Naval Oceanographic
and Atmospheric
Research Laboratory

Technical Note 193
November 1991

E040167

Documentation for the Semi-Automated Mesoscale Analysis System 1.1

S. H. Peckinpaugh
Remote Sensing Branch
Ocean Sensing and Prediction Division
Ocean Science Directorate

Approved for public release; distribution is unlimited. Naval Oceanographic and Atmospheric Research Laboratory, Stennis Space Center, Mississippi 39529-5004.

ABSTRACT

Several software modules have been developed for or by the Naval Oceanographic and Atmospheric Research Laboratory to automate the analysis and interpretation of satellite infrared imagery in the Gulf Stream region of the Northwest Atlantic Ocean. The modules have been integrated to form the Semi-Automated Mesoscale Analysis System version 1.1 (SAMAS). This document provides the information required for an analyst to use SAMAS 1.1. This document provides descriptions of how the modules of SAMAS 1.1 work, and the user inputs and outputs are described. This document also provides information on the location of source code and special data files required for SAMAS 1.1.

ACKNOWLEDGMENTS

SAMAS 1.1 has been made possible by the hard work of many individuals. The system was supported by the Office of Naval Technology, Program Element 0602435N, Program Manager CDR Lee Bounds. Major modules of SAMAS 1.1 were developed under the Data Assimilation and Analysis Technology Project of this program element, Ron Holyer Project Manager. The Expert System was developed by Perceptics Corporation/University of Tennessee, Matthew Lybanon contract technical officer (Contract No. N00014-87-C-6001). The labeling software was developed by Louisiana State University, N. Krishnakumar and S. Sitharama Iyengar (Contract No. N00014-88-K-6002). Planning Systems Incorporated developed the Complex Empirical Orthogonal Function interpolation of the Gulf Stream, Eugene J. Molinelli and Michael J. Flanigan (Contract No. N66604-86-D-0120). Bobby Grant of Sverdrup Technology, Inc. provided much of the processing for the testing and validation of the SAMAS 1.1.

The mention of commercial products or the use of company names does not in any way imply endorsement by the U.S. Navy or NOARL.

Table of Contents

DOCUMENTATION FOR THE SEMI-AUTOMATED MESOSCALE ANALYSIS SYSTEM 1.1

1. Introduction

Several software modules have been developed for or by the Naval Oceanographic and Atmospheric Research Laboratory (NOARL) to automate the analysis and interpretation of satellite infrared (IR) imagery in the Gulf Stream region of the Northwest Atlantic Ocean. The modules have been integrated to form the Semi-Automated Mesoscale Analysis System version 1.1 (SAMAS), see figure 1. The SAMAS starts with a satellite IR image, can be a warmest pixel composite, with an optional cloud mask. From these an edge image is created. The edges are labeled by a relaxation method using the original image and a previous Gulf Stream and eddy analysis that has optionally been progressed in time by the Oceanographic Expert System. Once the edges are labeled, then SAMAS has modules to define the Gulf Stream using complex empirical orthogonal functions (CEOF) and a circular Hough Transform is used to define the eddies. The Expert System can be used to progress eddies from the previous analysis where cloud cover prevents a clear view of the features at the current time. This is done to eliminate the loss of eddies from one analysis to the next. The new Gulf Stream, the new eddies, the Expert System eddies, and other frontal edges can be viewed and edited by an analyst using a set of 3 special purpose image editors. The final output of the system is a chart of locations of mesoscale features for the input image.

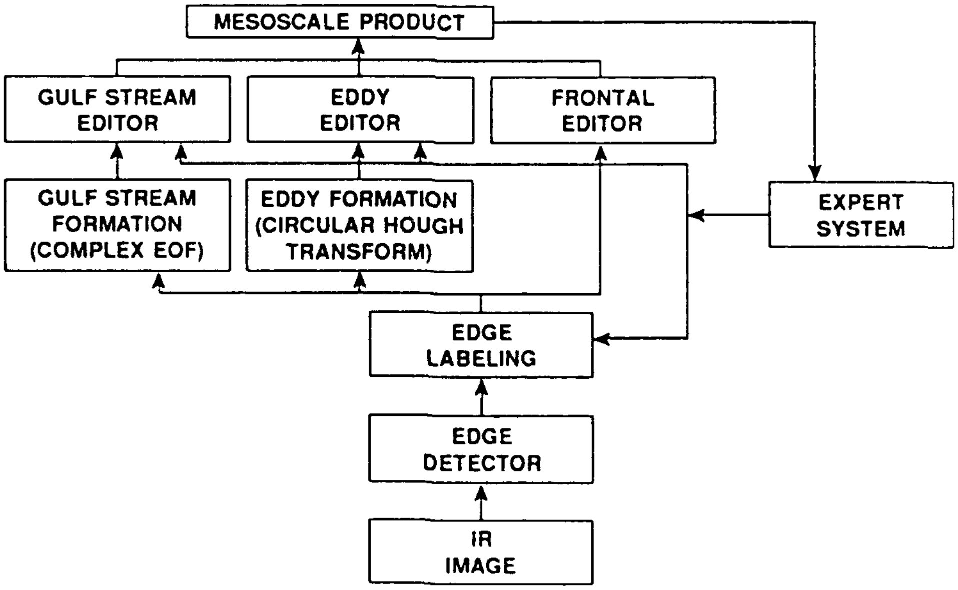

Figure 1. A functional block diagram of SAMAS 1.1.

2. SAMAS Driver and Functions

A driver has been developed to link the modules of the SAMAS into a user friendly menu-driven system. The system is set to run on the VAX 8800/IIS image processing system at the Remote Sensing Branch of NOARL. The interactive editors used by the analyst are run under the IIS image processing system even though they use the data formats designed for SAMAS. The SAMAS driver has the following menu:

1. Run User DE (Reformat images, Display images, Edit edges)
2. Run User BE (Reformat images, Display images, Edit edges)
3. Dump Image Values
4. Enter List - Eddy
5. Enter List - Gulf Stream
6. Run Expert System
7. Create Edge Image (Cluster Shade)
8. Label Edge Image (Relaxation method)
9. Run CEOF
10. Run Hough - Eddy Detect
11. Apply Image Mask
12. Execute VAX command
13. Quit (EXIT Menu)

The analyst can access the SAMAS menu by running the following executable:

```
Vax$User1:[Peckinpaugh.SAMAS.Screen]Driver.Exe
```

To prevent excess typing the analyst may choose to set a symbol, perhaps in the login.com file, as follows:

```
SAMAS == "$ Vax$User1:[Peckinpaugh.SAMAS.Screen]Driver.Exe"
```

If this is done, then the user can simply type the command SAMAS. To use this menu the analyst moves the cursor to the desired entry using the UP and DOWN arrow keys or by typing the entry number and RETURN. Once the cursor is at the desired entry RETURN will start the desired module. The menu list is circular and uses more than one terminal screen. When a module has finished the user will be urged to hit RETURN, this is done to allow the user to read any information printed by that module. The menu options are described below.

2.1 Run User DE or BE

A special IIS image processing user DE and BE have been developed for SAMAS. The user can select this option from the menu as item 1 or 2, respectively, or one of the following commands will work just as well:

```
UserDe Vax$User1:[Peckinpaugh.SAMAS.Exe]
UserBe Vax$User1:[Peckinpaugh.SAMAS.Exe]
```

These user versions contain modules to reformat IIS image files into SAMAS image files, display SAMAS image files on the IIS image processing display system, and display SAMAS graphics onto images on the IIS image processing display system. Also, included here are the SAMAS interactive editors for the Gulf Stream, eddies, and frontal edges. A list of the available functions is as follows:

* system'allocate
* system'directory
* system'quit

* m75'annotate
* m75'constant
* m75'directory
* m75'display
* m75'feedback
* m75'pause
* m75'save
* m75'select
* m75'tempzoom
* m75'tlm

* cpu'dump_image

* m75'graphics'blank
* m75'graphics'feedback
* m75'graphics'off
* m75'graphics'on
* m75'graphics'save
* m75'graphics'write

```
        m75'gulf_stream'edit

        m75'frontal'edit

        m75'eddy'edit

        cpu'samas'image_display
        cpu'samas'image_format
        cpu'samas'image_graphics

   *    system'command'list
```

For a description on running DE and BE see reference 1. For a complete description of the IIS modules listed above (marked with *) see reference 2.

2.1.1 CPU'SAMAS'Image_Display

This function is used to display SAMAS image files on the IIS model 75 display screen. The default suffix for a SAMAS image file is '.img'. The user supplies the input SAMAS image file name and an output display image name. The user may choose to display only a subsection or subsampled version of the image. Prompts are given for starting sample, number of samples, starting line, number of lines, and sampling increment. If no values are entered for the number of samples and lines, then either all samples and lines are displayed or the maximum that can be contained on the image screen (512 x 512). Images that are less than the size of the screen will be centered. The sampling increment will be applied to both the samples and lines dimensions. A sampling increment of 1 will skip no lines or samples, a sample increment of 2 will skip every other line and sample. The program prompts for this module are as follows:

Input SAMAS image to be displayed?
The user is to input the file name of the image to be displayed. The file must be in the SAMAS image format.

Output display image?
The user is to input the display image name to be used.

Starting_Sample: Starting sample of the input image to be displayed (def: 1)?
The user selects the first sample of the input image to be displayed. The default is the first sample of the image.

Number_Samples: Number of samples of the input image to be displayed?

The user selects the number of samples to be displayed. If no value is entered, then all samples after and including the starting sample are displayed or as many as will fit on the display screen.

Starting_Line: Starting line of the input image to be displayed (def: 1)?

The user selects the first line of the input image to be displayed. The default is the first line of the image.

Number_Lines: Number of lines of the input image to be displayed?

The user selects the number of lines to be displayed. If no value is entered, then all lines after and including the starting line are displayed or as many as will fit on the display screen.

SubSampling_Increment: Sub-sampling increment for the display of the input image (def: 1)?

The user selects the sampling increment to be applied to both the line and sample dimensions of the image being displayed. The default of 1 displays all lines and samples in the selected subsection.

2.1.2 CPU'SAMAS'Image_Format

This function is used to reformat IIS image files into SAMAS image files. The user is prompted for an input IIS image file name and an output SAMAS image file name, the default suffix for a SAMAS image file is '.img'. The user may choose to reformat only a subsectioned or subsampled version of the input image. Prompts are given for starting sample, number of samples, starting line, number of lines, and sampling increment. If no values are entered for the number of samples and lines, then all samples and lines after and including the starting coordinates are written to the output file. The sampling increment will be applied to both the samples and lines dimensions. The user is prompted for map registration information. These parameters are for the input IIS image file, prior to any subsectioning or subsampling. These values, North limit, West limit, and Degree/pixel, will be adjusted for the subsectioned/subsampled SAMAS image to be created. Adjusted values for the map registration will be written to the SAMAS image header, along with the number of samples, number of lines, data type, and map projection type. The program prompts are as follows:

Input image to be converted?

The user enters the file name for the IIS image to be converted to a SAMAS image file format.

Output SAMAS image?
The user enters the desired output SAMAS image file name.

Starting_Sample: Starting sample of the input image to be reformatted (def: 1)?
The user selects the first sample of the input image to be included in the output image. The default is the first sample of the image.

Number_Samples: Number of samples in the input image to be reformatted?
The user selects the number of samples to be included in the output image. If no value is entered, then all samples after and including the starting sample are used.

Starting_Line: Starting line of the input image to be reformatted (def: 1)?
The user selects the first line of the input image to be included in the output image. The default is the first line of the image.

Number_Lines: Number of lines in the input image to be reformatted?
The user selects the number of lines to be included in the output image. If no value is entered, then all lines after and including the starting line are used.

SubSampling_Increment: Sub-sampling increment for the input image to be reformatted (def: 1)?
The sampling increment to be applied to both the line and sample dimensions of the image being reformatted. The default of 1 uses all lines and samples in the specified subsection.

North_Latitude: North limit for the input image map projection?
This parameter defines the North latitude limit of the input IIS image being reformatted. This value will be adjusted automatically for the user specified starting line.

West_Longitude: West limit for the input image map projection?
This parameter defines the West longitude limit of the input IIS image being reformatted. This value will be adjusted automatically for the user specified starting sample.

Degree_Pixel_Ratio: Degree/pixel for the input image map projection?
This parameter defines the Degree/Pixel ratio used to map register the input IIS image. This value will be adjusted automatically for the user specified sampling increment.

Projection: Type of the input image map projection (Def: Rectilinear)?
This parameter defines the type of map projection; rectilinear or mercator, used to map register the input IIS image.

2.1.3 CPU'SAMAS'Image_Graphics

This function is used to display SAMAS image files as graphics on the IIS model 75 display screen. The default suffix for a SAMAS image file is '.img'. The user supplies the input SAMAS image file name and which planes of the image are to be displayed. The user may choose to display only a subsectioned or subsampled version of the image. Prompts are given for starting sample, number of samples, starting line, number of lines, and sampling increment. If no values are entered for the number of samples and lines, then either all samples and lines are displayed or the maximum that can be contained on the image screen (512 x 512). Images that are less than the size of the screen will be centered. The sampling increment will be applied to both the samples and lines dimensions. The program prompts are as follows:

Input SAMAS image to be displayed?
The user is to input the file name of the image to be displayed. It must be in the SAMAS image format.

Planes: The planes of the input image to be displayed (def: all)?
The user selects the bit planes of the input SAMAS image that are to be displayed. The valid range is 0 - 7, for the 8 bits of a byte value. Multiple values can be entered.

Starting_Sample: Starting sample of the input image to be displayed (def: 1)?
The user selects the first sample to be displayed of the input image. The default is the first sample of the image.

Number_Samples: Number of samples of the input image to be displayed?
The user selects the number of samples to be displayed. If no value is entered, then all samples after and including the starting sample are displayed or as many as will fit on the display screen.

Starting_Line: Starting line of the input image to be displayed (def: 1)?
The user selects the first line to be displayed of the input image. The default is the first line of the image.

Number_Lines: Number of lines of the input image to be displayed?
The user selects the number of lines to be displayed. If no value is entered, then all lines after and including the starting line are displayed or as many as will fit on the display screen.

SubSampling_Increment: Sub-sampling increment for the displayed of the input image (def: 1)?

The sampling increment to be applied to both the line and sample dimensions of the image being displayed. The default of 1 displays all lines and samples from the specified subsection.

2.1.4 M75'Eddy'Edit

This function allows a user to graphically display and modify an eddy list file. Buttons [A3] and [B3] allow the user to enhance the background image. Button [A3] puts the user in TLM mode. The cursor is used to control the ramp of the Look-Up table (LUT) conversion of the input byte pixel values to the display pixel intensities. Button [B3] allows the user to reverse or negate the LUT.

This function uses 3 graphics planes. These 3 graphics planes will be initially blanked. The first plane is used for scratch--drawing new eddies and highlighting eddies for possible deletion. The second plane is used to display the saved eddies, those that are in the current eddy list. The third plane is used to display the optional input Gulf Streams. The user may choose to display other graphics in the remaining planes; frontal edges, maps, grids, etc. Using button [C3] the user can change the color of any of the 3 reserved planes or by defining a color can turn on other desired ancillary graphics planes. Button [D3] can be used to turn any of the graphics planes off.

If one or more Gulf Stream list files were specified, then each of the files is read and the Gulf Stream plotted. These files contain in the first record the number of latitude, longitude positions contained in the file. The remaining records contain the latitude, longitude positions and a source code value for each point making up the Gulf Stream. The code value will be ignored by this program. Each point of a Gulf Stream is marked in the Gulf Stream graphics plane and lines are drawn connecting the points.

If one or more old eddy list files were specified, then each of the eddy list files is read and each eddy drawn. Each of the eddies is displayed in the scratch plane and the user is prompted to "Keep or Delete?" that eddy. If the user selects to keep an eddy, then it is added to the new eddy list and drawn in the eddy save graphics plane. If a user selects to keep an eddy that overlaps an existing eddy, then the previously saved eddy will be deleted. No overlapping of eddies is allowed.

The parameters for starting sample, starting line, and sampling increment define what area of the first input image will be used. The map registration information is read from the image header record. If more than one input image is supplied, then each of the remaining input images are opened and the map registration information is checked against the first input image, for each either the starting sample, starting line, and sampling increment are used for that image or the image is determined to have already been adjusted for these values; thus, the image will be displayed without further subsectioning and subsampling. If neither of these conditions apply, then an error is generated for mapping mismatch and the function is terminated.

The user is given 4 options for creating new eddies for the eddy list. Buttons [A2], [B2], and [C2] allow the user to fix some part of the eddy and, then use the cursor to interactively change the size or position of the eddy. The eddy will be drawn in the scratch plane as the user modifies the cursor controlled parameter. Button [A2] fixes the eddy center position at the current cursor position. The cursor is, then used to interactively define the eddy radius. Button [B2] prompts the user for the desired kilometer radius. The cursor is, then used to interactively define the eddy center position. Button [C2] fixes the current cursor position as a point on the eddy circumference. The user is asked to select a second point using the cursor. This point will also remain fixed on the eddy circumference. The cursor is, then used to interactively define a third point on the eddy circumference. Button [B1] is used to save the interactively created eddy. The eddy will be erased from the scratch plane, drawn in the eddy plane, and added to the eddy list. Button [F1] is used to clear an eddy currently being created using button [A2], [B2], or [C2]. The eddy will be erased from the scratch plane. Eddies can also be added to the eddy list via button [D2]. The user is prompted for the eddy definition information. The first prompt is for eddy center latitude. If the user does not enter a value for this parameter, then prompting stops and no new eddy is created. If a center latitude is specified, then the user is prompted for center longitude, kilometer radius, eddy type (warm or cold), and source code. The eddy will be drawn in the eddy plane and added to the eddy list. Whenever an eddy is saved, which overlaps an existing eddy, the previously saved eddy will be deleted from the eddy list and erased from the eddy plane. No overlapping of eddies is allowed. The user will be prompted for more eddies until no value is entered for latitude.

Button [A1] is used to delete an existing eddy. The cursor must be on the eddy to be deleted. The eddy will be highlighted in the scratch plane and the user will be prompted for "Keep or Delete?". If delete is selected, then the eddy will be erased from the eddy plane and deleted from the eddy list.

Interactively created eddies initially have a label of U for Undefined. The user can move the cursor onto an existing eddy, then button [C1] can be used to prompt the user to define or redefine the eddy type as W for Warm, C for Cold, or undefined.

Button [D1] prints the eddy list in labeled column form on the user's terminal screen. The information printed for each eddy is screen center line position, screen center sample position, radius in pixels, center latitude, center longitude, radius in kilometers, source code, and type (warm or cold). If the cursor is currently on an existing eddy, then that eddy will be highlighted in the list. If the user is interactively creating an eddy, then the information for that eddy will appear at the end of the list.

Button [F3] causes an exit from the function. The newly created eddy list file will be written to the output file. There is a header record for the number of eddies in the file. After that, there is one record for each saved eddy in the eddy list file. The values that are written to define the eddy are as follows: center latitude position, center longitude position, kilometer radius, type code (Warm, Cold, or Undefined), and source code.

The user is provided a menu in the status plane of the model 75. The menu will appear as follows:

	A	B	C	D	F
3	Image TLM Toggle	Negate Image	Change graphics colors	Graphics plane Off	Exit Function
2	Draw eddy from fixed center	Draw Eddy from fixed radius	Draw Eddy from 2 fixed points	Enter new eddy	
1	Delete Eddy	Keep Eddy	Label WARM or COLD	Print Eddy info	Clear current Eddy draw

Menu: Eddy'Edit

The menu buttons are defined in more detail below:

A3 Puts the function into TLM mode. The user can then use the cursor to interactively modify the linear mapping of the byte pixel values of the background image into the screen intensity values.

B3 Causes the background image enhancement to be negated.

C3 The user is prompted for plane number and color weights to be applied to that plane. This can be used to change the color of the 3 active graphics planes of the function, or it can be used to turn on and color other desired graphic planes--like a grid or land mask.

D3 The user is prompted for the graphics plane to turn off. The plane can be one of the working planes of the function or a plane the user has enabled using button [C3].

F3 Exits the function and saves the newly created eddy list file.

A2 Allows the user to fix the eddy center at the current cursor position and, then move the cursor to interactively define the eddy radius. As the cursor is moved the new eddy is drawn in the scratch-graphics plane.

B2 The user is prompted to enter the desired kilometer radius for the eddy. The cursor is then used to interactively define the eddy center position. As the cursor is moved the new eddy is drawn in the scratch-graphics plane.

C2 The current cursor position becomes a fixed point on the eddy edge. The user is prompted to use the cursor to select a second fixed position on the eddy edge. The cursor is then used to interactively define a third point on the eddy edge. As the cursor is moved the new eddy is drawn in the scratch-graphics plane.

D2 This button allows the user to enter the values that define an eddy. These values are center latitude, center longitude, kilometer radius, source code, and type (warm or cold).

A1 If the cursor is on an existing eddy, then the eddy will be highlighted in the scratch plane and the user will be prompted for "Keep or Delete?".

B1 When a user has created an eddy interactively using buttons [A2], [B2], or [C2], this button is used to save the newly created eddy. The eddy will be erased from the scratch plane, drawn in the eddy plane, and added to the eddy list.

C1 If the cursor is on an existing eddy, then the user will be prompted for eddy type, Warm or Cold. The specified label will be assigned to that eddy.

D1 The eddy list is printed in labeled column form on the user's terminal screen. The information printed for each eddy is screen center line position, screen center sample position, radius in pixels, center latitude, center longitude, radius in kilometers, source code, and type (warm or cold). If the cursor is currently on an existing eddy, then that eddy will be highlighted in the list. If the user is interactively creating an eddy, then the information for that eddy will appear at the end of the list.

F1 Clears from the scratch plane any eddy being drawn by the user via button [A2], [B2], or [C2].

The user is prompted for the following information, the last 6 of these are conditional. If the user selects in prompt 8 not to define the graphic planes, then the default values will be used and the user will not see the last 6 prompts.

Input image(s)?
The user can enter from 1 to 10 input SAMAS image file names. These images will be used as background for this function. The displayed area of the image(s) will be centered on the image display screen.

Gulf Stream file(s)?
The user can enter up to 14, optional, input Gulf Steam list files. These Gulf Streams will be plotted in the Gulf Stream graphics plane.

Previous eddy list file(s)?
The user can enter up to 14, optional, input Eddy list files. The eddies from these files will be plotted in the scratch-graphics plane and the user will be prompted "Keep or Delete?" for each. The eddies selected for keep by the user will be plotted in the eddy graphics plane and added to the eddy list. If the user selects to keep an eddy that overlaps an eddy that has already been kept, the previously kept eddy will be deleted.

Output eddy list file?
The user must enter the file name for the output eddy list.

Starting sample for the input image(s)?
The user selects the first sample of the first input image to be displayed. To match the mapping of the first input image, the remaining images will be checked to determine if they should be displayed using this starting sample or the first sample of the image.

Starting line for the input image(s)?

The user selects the first line of the first input image to be displayed. To match the mapping of the first input image, the remaining images will be checked to determine if they should be displayed using this starting line or the first line of the image.

Sub-sampling increment for the input image(s)?

The user selects the sampling increment to be used to display the first input image. To match the mapping of the first input image, the remaining images will be checked to determine if they should be displayed using this sampling increment or a sample increment of 1.

Do you wish to define the graphic planes and their colors (Def: No)?

This parameter is given to determine if the user wishes to reply to the next six questions concerning the selection and coloring of the reserved graphics planes for this function.

Gulf Stream graphics plane (Def: 6)?

This prompt is used to define which graphics plane is to be used to plot the optional input Gulf Stream list files. The range is 0 - 7.

Gulf Stream color (R,G,B)?

This prompt is used to define the red, green, and blue weights for the Gulf Stream graphics plane. The range for these values is 0.0 - 1.0.

Eddy graphics plane (Def: 5)?

This prompt is used to define which graphics plane is to be used to draw selected eddies. The range is 0 - 7.

Eddy color (R,G,B)?

This prompt is used to define the red, green, and blue weights for the eddy graphics plane. The range for these values is 0.0 - 1.0.

Scratch-graphics plane (Def: 4)?

This prompt is used to define which graphics plane is to be used for drawing and highlighting eddies. The range is 0 - 7.

Scratch color (R,G,B)?

This prompt is used to define the red, green, and blue weights for the scratch-graphics plane. The range for these values is 0.0 - 1.0.

2.1.5 M75'Frontal'Edit

This function allows a user to display and modify a frontal edge image. An optional background image can be displayed. If no background image is specified, then a black background will be supplied. Buttons [A3] and [B3] allow the user to enhance the background image. Button [A3] puts the user in TLM mode. The cursor is used to control the ramp of the LUT conversion of the input byte pixel values to the display pixel intensities. Button [B3] allows the user to reverse or negate the LUT.

The edge image is read. Any nonzero values of the input edge image will be set to 1, or on, in the frontal graphics plane; others will be set to 0. The user selects the plane and color for the edge image display. This function uses 2 graphics planes. The second plane is the scratch plane. This plane is used to highlight segments being considered for deletion and display new segments the user is adding to the edge image. Using button [C3] the user can change the color of either of these 2 planes or color and turn on other desired ancillary graphics planes. Button [D3] can then be used to turn off any of the graphics planes.

The map registration of the input images is checked to determine if the starting sample, starting line, and sampling increment have already been applied to some of the images when compared one to another. If the map information matches for all of the input images, then the starting sample, starting line, and sampling increment will be applied to each of the input images. They will be displayed starting at the starting sample and line using the sampling increment. If it is determined that some of the input images have already been subsectioned and subsampled using the specified starting sample, starting line, and sampling increment, then these images will be displayed starting at the first sample and line using all pixels of the image. The remaining images will be displayed using the specified starting sample, starting line, and sampling increment. If the map information for the images does not match, even assuming that some have already been subsectioned and subsampled by the specified parameters, then an error is produced and the function terminates.

The user is given 2 methods of deleting segments from the edge image. The first method is to simply move the cursor over some point of the segment to be deleted and button [A1] will highlight the segment. The user can then go to another step or initiate deletion via button [A2]. A prompt will be given--"Keep or Delete?". If Delete is selected, then the segment will be erased from the frontal graphics plane. For the purpose of removing parts of segments or groups of small segments the user can select areas of the edge image by creating a polygon, button [B1] starts the process. Points of the polygon are created by selecting points via button [C1] and, if necessary points can be deleted using buttons

[C2] which deletes the previous point selected or [D2] which deletes all selected points. The first and the last points selected for the polygon will be connected. The polygon and all edges contained within the polygon are highlighted. The user can initiate deletion via button [B2]. A prompt will be given--"Keep or Delete?". If Delete is selected, then the segments contained within the polygon will be erased from the frontal graphics plane.

The user can add segments to the edge image. The input Dilate_Steps defines the width to be used for adding segments. A dilation of 0 gives 1 pixel wide lines. Line width will be 1 plus twice the number of dilation steps. Button [D1] initiates the selection of points to define the segment. The user adds points to the line using the cursor position and button [C1]. Selected points can be deleted using buttons [C2] which deletes the previous point selected or [D2] which deletes all selected points. When the user exits this mode a prompt will be given--"Keep or Delete?". Keep will cause the segment to be drawn in the frontal graphics plane.

Button [F3] causes an exit from the function. The modified edge image will then be written to the output file.

The user is provided a menu in the status plane of the model 75. The menu will appear as follows:

	A	B	C	D	F
3	Image TLM Toggle	Negate Image	Change graphics colors	Graphics plane Off	Exit Function
2	Delete segment	Delete polygon	Delete last point	Delete all points	
1	Select old segment	Select polygon	Select point	Add new segment	

Menu: Frontal’Edit

The menu buttons are defined in more detail below:

A3 Puts the function into TLM mode. The user can then use the cursor to interactively modify the linear mapping of the byte pixel values of the background image to the screen intensity values.

B3 Causes the background image enhancement to be negated.

C3 The user is prompted for plane number and color weights to be applied to that plane. This can be used to change the color of the 2 active graphics planes of the function, or it can be used to turn on and color other desired graphic planes--like a grid or land mask.

D3 The user is prompted for the graphics plane to turn off. The plane can be one of the working planes of the function or a plane the user has enabled using button [C3].

F3 Exit the function and save the newly modified frontal edge image.

A2 If a segment has been selected, highlighted, using the cursor and button [A1], then the user is prompted for "Keep or Delete?". If the user selects Delete, then the segment will be removed from the frontal edge image.

B2 If segments have been selected, highlighted, by creating a polygon, then the user is prompted for "Keep or Delete?". If the user selects Delete, then the segments will be removed from the frontal edge image.

C2 This button is only active if the user has opted via button [D1] to add a segment to the frontal edge image or has opted via button [B1] to select a polygon. If any points have been selected for either the polygon or new segment, then the point that was selected last will be deleted.

D2 This button is only active if the user has opted via button [D1] to add a segment to the frontal edge image or has opted via button [B1] to select a polygon. If any points have been selected for either the polygon or new segment, then the selected points will be deleted.

A1 If the cursor is on a frontal edge, then that edge will be highlighted in the scratch plane.

B1 Allows the user to define a polygon. The first point selected will be connected to the last point selected. Any frontal edges contained within the polygon will be highlighted in the scratch plane.

C1 This button is only active if the user has opted via button [D1] to add a segment to the frontal edge image or has opted via button [B1] to select a polygon. The cursor is used to select new points on the polygon edge or to add points to the new frontal edge being drawn.

D1 Allows the user to define a new frontal edge.

The user is prompted for the following information:

Input image(s)?

The user may enter from 1 to 10 input SAMAS image file. The first image named is the input frontal image. This is the image that the user will be editing. The other, optional, input images are background images.

Output image?

The user must enter the file name for the output frontal edge image.

Starting_Sample: Starting sample of the input image display (def: 1)?

The user selects the starting sample used to subsection some or all of the input images. If all of the input images have the same map registration, then starting sample is used to start the display for all of them. If some of the input images have already been subsectioned using this starting sample, then these will be displayed starting at sample 1. While the others will be displayed using the starting sample specified.

Starting_Line: Starting line of the input image display (def: 1)?

The user selects the starting line used to subsection some or all of the input images. If all of the input images have the same map registration, then starting line is used to start the display for all of them. If some of the input images have already been subsectioned using this starting line, then these will be displayed starting at line 1. While the others will be displayed using the starting line specified.

SubSampling_Increment: Sub-sampling increment for the input image (def: 1)?

The user selects the subsampling increment used to subsample some or all of the input images. If all of the input images have the same map registration, then the subsampling increment is used to subsample the display of all of them. If some of the input images have already been subsampled using this increment, then these will be displayed using all samples of the image. While the other images will be displayed using the subsampling increment specified.

Dilate_Steps: Number of dilate steps for new segments (def: 0)?

The number of dilate steps determines how thick new segments of the frontal edge image will be. Segments with no dilation are 1 pixel thick. For each step of dilation a pixel width is added to each side of the new segment. Example: 1 step of dilation produces a 3 pixel wide line.

Frontal_Plane: Graphics plane for frontal image (def: 6)?

Frontal_Plane is the graphics plane where the frontal edges are drawn. These edges, after editing, will be written to the output frontal edge image.

Frontal_Colors: Red, green, and blue weights (def: 1.0 0.0 0.0)?

This parameter defines the red, green, and blue color weights for the color definition of the Frontal_Plane. The valid range for these values is 0.0 - 1.0.

Scratch_Plane: Scratch-graphics plane (def: 5)?

Scratch_Plane is the graphics plane where the user draws new edges and highlights existing edges for possible deletion.

Scratch_Colors: Red, green, and blue color weights (def: 0.0 1.0 0.0)?

This parameter defines the red, green, and blue weights for the color definition of the Scratch_Plane. The valid range for these values is 0.0 - 1.0.

2.1.6 M75'Gulf_Stream'Edit

This function allows a user to graphically display input Gulf Steam list file(s), a frontal edge image, and a background image, and then use these to create a new Gulf Stream list file. Buttons [A3] and [B3] allow the user to enhance the background image. Button [A3] puts the user in TLM mode. The cursor is used to control the ramp of the LUT conversion of the input byte pixel values to the display pixel intensities. Button [B3] allows the user to reverse or negate the LUT. Button [F2] allows the user to zoom the background image along with the graphics. Each [F2] button selection cycles to the next choice: 2X, 4X, 8X, reset to no zoom, and back to 2X.

This function uses 4 graphics planes. These 4 graphics planes will be initially blanked. The first plane is used for scratch--drawing new segments for the Gulf Stream and highlighting previously saved segments for possible deletion. The second plane is used to display the edited Gulf Stream, those positions that are in the current Gulf Stream list. The third plane is used to display the optional input Gulf Stream(s). The fourth plane is used to display the input frontal edge image. Nonzero pixels of the input frontal edge image are set in this plane. The user may choose to display other graphics in the remaining planes--maps, grids, etc. Using button [C3] the user can change the color of any of the 4 reserved planes or by defining a color turn on other desired ancillary graphics planes. Button [D3] can be used to toggle any of the graphics planes Off/On.

If one or more Gulf Stream list files were specified, then each of the files is read and the Gulf Stream drawn. These files contain in the first record the number of latitude, longitude positions contained in the file. The remaining records contain the latitude, longitude positions and a source code value for each point making up the Gulf Stream. The code value along with the flags Connect_I and Connect_All determine which points of the input Gulf Streams are drawn as single points and which points are connected with line segments. If Connect_I is Yes, then positions with a source code of 'I' will be drawn connected with line segments. If Connect_All is Yes, then all positions of the input Gulf Streams will be connected with line segments. If both parameters for connect are no, then the Gulf Stream will be just plotted point positions.

The user enters the desired starting sample, starting line, and sampling increment. If a background image is provided, then these values are used to subsection and subsample the image for display. The image will be displayed starting at the starting sample and line and every sample increment sample and line will be displayed. If a frontal image and a background image are specified, then the map registration for the 2 images is compared to determine if the frontal image has already been subsectioned and subsampled by these parameters. If it has been, then it will be displayed starting at the first sample and line using all samples and lines. If it has not been or if no background image was specified, then the image will be subsectioned and subsampled by the starting sample, starting line, and sampling increment. If the frontal image map information does not match the background image map information for either of these cases, then an error is generated and the function terminates.

The user is given 3 button options for adding to the new Gulf Stream. Button [D2] is used to simply select the current cursor position as the next point. Where the new point will be added to the Gulf Stream depends on how the cursor is attached to the Gulf Stream. As the cursor is moved the connection of the cursor to the Gulf Stream is drawn in the scratch-graphics plane. Button [B2] is used to select points from the frontal edges to be included in the Gulf Stream. The cursor must be on a frontal edge. If the frontal edges are more than 1 pixel wide, then the edge will be thinned and the single pixel wide edge will be traced from the point closest to the current cursor position. At any given point on a thinned edge 1, 2, or at most 3 directions are available for tracing. For the selected direction the longest possible trace is done. As the trace is being done if a choice of possible branches is encountered, then the longest branch is selected. The trace will use every pixel position along the thinned edge, unless the user via button [D1] has changed the trace increment. If the user has defined the trace increment, then every trace increment pixel along the thinned edge will be used. The trace is highlighted in the scratch-graphics plane along with the connections to the current Gulf Stream. The user is prompted: "Keep or Delete?". If the user keeps the trace, then it will become part of the current Gulf

Stream. If the user does not keep the trace, then the next possible direction from the cursor is processed; if all possible directions from the current cursor position have been rejected, then menu control is restored. The last method for adding points to the Gulf Stream is to trace an input Gulf Stream. Button [B1] is used for tracing an input Gulf Stream. The closest point on the input Gulf Stream to the current cursor position is used as the start of the trace. If the current cursor position is at one end of the Gulf Stream, then only 1 trace will be processed. The trace will include the entire input Gulf Stream. The trace will be highlighted in the scratch-graphic plane with connections to the new Gulf Stream; the user will be prompted: "Keep or Delete?". If the user Keeps the trace, then it will become part of the current Gulf Stream. Menu control is restored. If the current cursor position is not at one of the endpoints of the input Gulf Stream, then a trace will be highlighted with connections for the start of the input Gulf Stream to the point of the input Gulf Stream closest to the current cursor position. The user will be prompted: "Keep or Delete?". If the user Keeps the trace, then it will become part of the current Gulf Stream and menu control is restored. If the user does not keep the trace, then the remaining direction is processed. The trace will be highlighted with connections for the remainder of the input Gulf Stream. If the user selects to Keep this trace, then it will become part of the current Gulf Stream. Menu control is restored.

Some points of the new Gulf Stream will be connected with segments others will be shown as single point positions. The user controls the code assigned to selected points. If 2 consecutive points have a code of 'I', then they will be drawn with a connecting segment. Points with a code of 'O' will be shown as single-point positions. When keeping a trace of an input Gulf Stream, if the points are drawn connected for the input Gulf Stream, then they will have a code of 'I' when saved; if the points are drawn as single point positions, then they will have a code of 'O' when saved. The user determines at the initial input which points of the input Gulf Stream will be connected. Button [A2] toggles on/off the connect flag. A message on the user's terminal screen shows the current state of the flag. When selecting points or tracing edges, if this flag is TRUE, then saved points will have a code of 'I'; else, saved points will have a code of 'O'. Button [F1] can be used to temporarily show all points of the edited Gulf Stream connected. This is shown in the scratch-graphics plane, until cursor movement or button input from the user turns it off.

Button [A1] allows the user to define where new points will be added or deleted from the new Gulf Stream. The user can select either the end positions or between two points. As the cursor is moved the connection of the position to the new Gulf Stream is drawn in the scratch-graphics plane.

The user is given 2 button options for deleting undesired points from the new Gulf Stream. Button [C2] allows the user to delete single points. If the cursor is connected to the end of the new Gulf Stream, then the endpoint to which it is attached will be deleted. If the cursor is attached between 2 points of the Gulf Stream, then of these 2 points the closest to the current cursor position will be deleted. Button [C1] allows the user to select a continuous segment of points of the new Gulf Stream to delete. The point to which the cursor is attached will be the start or first endpoint. The user moves the cursor to select the second endpoint. Button [A1] is active so that the user can reselect the fixed endpoint. Button [F2] is active to reset the zoom. The segment being selected is highlighted in the scratch-graphics plane as the user moves the cursor. Any button response other than [A1] or [F2] will end the segment selection. The user is prompted: "Keep or Delete?". If the user selects delete, then the segment will be deleted from the new Gulf Stream. Menu control is restored.

Button [F3] causes an exit from the function. The newly created Gulf Stream list file will be created. The first record of the output file contains the number of points defining the Gulf Stream. The remaining records, one for each point position, contain latitude, longitude, and a source code of either 'I' or 'O'.

The user is provided a menu in the status plane of the model 75. The menu will appear as follows:

	A	B	C	D	F
3	TLM	Negate Image	Change graphics colors	Graphics plane Off/On	Exit Function
2	Connect points On/Off	Auto-trace frontal edge	Delete point	Select point	Zoom Increment/ Reset
1	Attach Cursor	Auto-trace input Gulf Stream	Delete segment	Change trace increment	Show connect all

Menu: Gulf_Stream'Edit

The menu buttons are defined in more detail below:

A3 Puts the function into TLM mode. The user can then use the cursor to interactively modify the linear mapping of the byte pixel values of the background image to the screen intensity values.

B3 Causes the background image enhancement to be negated.

C3 The user is prompted for plane number and color weights to be applied to that plane. This can be used to change the color of the 4 active graphics planes of the function, or it can be used to turn on and color other desired graphic planes like--a grid or land mask.

D3 The user is prompted for the graphics plane to toggle off/on. The plane can be one of the working planes of the function or a plane the user has enabled.

F3 Exit the function and save the output Gulf Stream list.

A2 Enable/disable connect points. Points selected when this parameter is enabled will be assigned a code of 'I'. Consecutive points with 'I' labels will be connected. Points selected when this parameter is disabled will be assigned a code of 'O'.

B2 Trace the frontal edge image starting at the cursor position. The trace will go in all available directions from the cursor position, until the user selects to keep a trace or all directions have been rejected. The longest path for each case or direction is highlighted and the user is prompted to "Keep or Delete?" each of the choices.

C2 Delete the nearest point to which the cursor is currently attached.

D2 Select the cursor position as a new point of the Gulf Stream.

F2 Zoom the image by 2, 4, or 8 and, then reset to no zoom, cycle to next choice for each button response.

A1 Attach the cursor to a new point of the Gulf Stream. The cursor can be attached to either of the endpoints or between 2 interior points of the edited Gulf Stream.

B1 Trace the old Gulf Stream. Each available direction will be traced and highlighted, until a trace is kept or all directions have been rejected. The user is prompted to "Keep or Delete?" the trace.

C1 Allows the user to delete a segment of the Gulf Stream. The point at which the cursor is attached is the first endpoint. Button [A1] is active; thus, the first endpoint can be changed. The cursor movement then controls the second endpoint selection. Buttons other than [A1] and [F2] will cause the user to be prompted "Keep or Delete?" the highlighted segment.

D1 The user is prompted for a new trace increment. This is used when tracing the frontal edges. The increment determines the point spacing for tracing the frontal edges.

F1 Temporarily draws lines in the scratch plane connecting all points of the new Gulf Stream. Any cursor movement or button input other than [F1] will erase the lines.

The user is prompted for the following information:

Input background image?
The user can enter an optional input background image. Either a background or frontal image must be specified.

Input frontal edge image?
The user can enter an optional input frontal image. Either a background or frontal image must be specified.

Input_Gulf_Stream: Input Gulf Stream list file(s) (opt.)?
The user can enter up to 10 optional input Gulf Stream files.

Connect_I: Connect points labeled 'I' in the Gulf Stream list file (Def: Yes)?
This prompt will only be given if an input Gulf Stream file has been specified. If Connect_I is Yes, then consecutive points of the input Gulf Streams having a code of 'I' will be connected. If Connect_I and Connect_All have been set to No, then the input Gulf Streams will be plotted points with no connecting lines.

Connect_All: Connect all points in the Gulf Stream list file (def: No)?
This prompt will only be given if an input Gulf Stream file has been specified. If Connect_All is Yes, then consecutive points of the input Gulf Streams will be connected. If Connect_I and Connect_All have been set to No, then the input Gulf Streams will be plotted points with no connecting lines.

Output_Gulf_Stream: Output Gulf Stream list file?
The user must enter the file name for the output Gulf Stream list file.

Starting_Sample: Starting sample of the input image to display (def: 1)?
The user can enter the sample to start the display of the background image and/or the frontal image. If a background image has been specified and the frontal image has already been subsectioned by the starting sample prior to this function, then it will be displayed starting at the first sample.

Starting_Line: Starting line of the input image to display (def: 1)?

The user can enter the line to start the display of the background image and/or the frontal image. If a background image has been specified and the frontal image has already been subsectioned by the starting line prior to this function, then it will be displayed starting at the first line.

SubSampling_Increment: Sub-sampling increment for the input image (def: 1)?

The user can enter the subsample increment to use for the display of the background image and/or the frontal image. If a background image has been specified and the frontal image has already been subsampled by the sampling increment prior to this function, then it will be displayed using all samples and lines.

2.2 Dump Image Values

The user can select this option from the menu as item 3, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]dump_image" will work just as well. This function is used to dump the header information and data values from SAMAS image files. The user provides an input file name. The program prints the header information: number of samples, number of lines, data type, North latitude limit, West longitude limit, degree/pixel ratio, and type of map projection. The user will then be asked for "starting sample and line - ". The user must enter 2 values separated by a blank. Starting at the specified sample and line, 10 lines of 10 samples will be printed for the user. The user will be reprompted for "starting sample and line - " and the data printed, until 0 values are entered for the prompt. The program prompts are as follows:

Input image name -

The user enters the name of the input SAMAS image file.

starting sample and line -

The user enters the sample and line coordinates of the upper, left corner of the 10 x 10 pixel box of data to be dumped. The user will be reprompted for these values until values of 0 are input to end the program run. The values should be separated by a blank, not a comma.

An example run of this program follows:

```
Input image name -  Test.Img
Nsamp - 70
Nline - 70
Dtype - Byte
North - 45.000000
West - -75.000000
DegPixel - 0.025000
Proj - Merc

starting sample and line - 1 1

Rec 1 samp 1
     183  175  180  182  169  122  182  192  183  173
     171  186  174  125  168  179  179  186  171  170
     145  183  162  159  147  182  187  175  179  169
     184  188  171  167  168  179  183  177  149  160
     188  192  174  179  183  186  186  170  155  170
     187  158  175  184  196  186  177  173  170  172
     181  184  184  187  183  177  177  162  161  169
     186  184  183  182  175  181  162  172  135  210
     190  182  183  185  178  191  162  173  180  124


starting sample and line - 0 0
```

2.3 Enter List - Eddy

The user can select this option from the menu as item 4, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]eddy_enterlist" will work just as well. This function allows the user to create an eddy list file. The user can optionally input an eddy list file. The eddies from the input file will be written to the new output eddy list. The user can select to enter the eddy radius values in either kilometers or nautical miles. The user is prompted for a 2 character ASCII source code. This code is strictly for the user's reference. Some eddy creating modules of SAMAS output source codes unique for the function, example: ED for Eddy Detect. The user is prompted for latitude, longitude, radius, and eddy type (Warm, Cold, or Undefined) until no value is entered for the prompt. Each eddy will have the previously entered source code. The user is prompted again for a source code. If no value is entered, then the new eddy list file is created from the data already entered by the user. If a new source code is entered, then the user is again prompted for more eddy information until none is entered. This process continues until no value is entered for source code. As eddies are entered into the list, they are checked against the eddies already contained in the list. If a new eddy is found to overlap any of the eddies already in the list, then the new eddy will be kept and the other eddy or eddies will be deleted from the list. No overlapping of eddies is allowed. The program prompts are as follows:

Input File -

The user is prompted for an optional input eddy list file. If a file is supplied, then the eddies of this file are written to the output list and the new eddies to be entered by the user will be appended to the list. If a new eddy overlaps an existing eddy, then the existing eddy will be deleted from the list.

Output File -

The user is prompted for an output eddy list file. This file will contain the newly created eddy list.

Radius values will be KM (def: K) or Nautical Miles (N) -

The user is prompted for the unit of measure for the eddy radius values to be entered. If the values are in nautical miles, then they will be converted to kilometers.

Enter Source Code -

The user is prompted for a 2 character ASCII source code. This code is used to define the origin of the eddy information. This code is used for all eddies entered until a new code is entered. If the user enters no code, then the output file is written and the program ends.

Enter Lat, Lon, Rad, Type -

The user enters these values along with the source code to define an eddy. The user enters the latitude, longitude, radius (in the units specified KM or Nautical miles), and type (Warm, Cold, or Undefined). If the user enters no values at this prompt, then the prompt for source code is given.

An example run of this program follows:

```
Input File -
Output File - Apr1.Eddy
Radius values will be KM (def: K) or Nautical Miles (N) -
Enter Source Code -  OC
Enter Lat, Lon, Rad, Type - 39.3, -70.3, 40, W
Enter Lat, Lon, Rad, Type -  40.0, -65.6, 39, W
Enter Lat, Lon, Rad, Type -
Enter Source Code -  GP
Enter Lat, Lon, Rad, Type - 35.2, -67.2, 45, C
Enter Lat, Lon, Rad, Type -
Enter Source Code -

     Latitude    Longitude     Radius      Type    Source
 1     39.300      -70.300     40.000         W        OC
 2     40.000      -65.600     39.000         W        OC
 3     35.200      -67.200     45.000         C        GP
```

2.4 Enter List - Gulf Stream

The user can select this option from the menu as item 5, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]GS_enterlist" will work just as well. This function allows the user to create a Gulf Stream list file. The user can optionally input a Gulf Stream list. The positions from this file will be written to the new output Gulf Stream list file. The user is prompted for a single character ASCII source code. This source code defines the origin of the data. The codes used are 'I' for image, 'A' for altimeter, and 'O' for other. Points with a code of 'I' are expected to be points of a continuous stream, instead of isolated points. The user is prompted for latitude and longitude until no value is entered for the prompt. Each position will have the previously entered source code. The user is prompted for a new source code. If no value is entered, then the new Gulf Stream list file is created. If a new source code is entered, then the user is again prompted for more Gulf Stream positions until no value is entered. This process continues until no value is entered for source code. The program prompts are as follows:

Input File -
The user is prompted for an optional input Gulf Stream list file. If a file is supplied, then the positions from this file are written to the output list and the new positions entered by the user will be appended to the list.

Output File -
The user is prompted for an output Gulf Stream list file. This file will contain the newly created Gulf Stream list.

Enter Source Code -
The user is prompted for a single character ASCII source code. This code is used to define the origin of the position information. This code is used for all positions entered until a new code is entered. If the user enters no code, then the output file is written and the program ends.

Enter Lat, Lon -
The user enters the next latitude, longitude position of the Gulf Stream. If the user enters no values at this prompt, then the prompt for source code is given.

An example run of this program follows:

```
Input File -  Apr1.NW
Output File -  Apr1_2.NW
Enter Source Code -  I
Enter Lat, Lon -  26.8, -79.9
Enter Lat, Lon -  27.3, -80
Enter Lat, Lon -  28.4, -80.4
Enter Lat, Lon -  29.4, -80.5
Enter Source Code -
```

2.5 Expert System

The user can select this option from the menu as item 6, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]expert" will work just as well. This function accepts as input any or all of the following: a Gulf Stream North wall, a Gulf Stream South wall, and an eddy list. At least one of the files must be provided. If a North wall is provided but no South wall, then a South wall will be created of the same shape and 100 km South of the North wall. If a South wall is provided but no North wall, then a North wall will be created of the same shape and 100 km north of the South wall. If neither wall is provided, then the default will be used for both. The user supplies a value for number of days. The North wall, South wall, and eddies will be progressed in time the number of days specified. The user has the option of keeping any or all 3 of the progressed lists; North wall, South wall, or eddy. Gulf Stream positions output by this function will have a source code of 'O'. Eddies output by this function will have a source code of 'ES'. The program prompts are as follows:

name of UGS file =
The user is prompted for the input upper Gulf Stream file (North wall). This file should be in the SAMAS format. The header record contains the number of positions. The remaining records contain the latitude, longitude, and source code for the Gulf Stream positions. The program does not use the source code.

name of LGS file =
The user is prompted for the input lower Gulf Stream file (South wall). This file should be in the SAMAS format. The header record contains the number of positions. The remaining records contain the latitude, longitude, and source code for the Gulf Stream positions. The program does not use the source code.

name of eddy file =
The user is prompted for the input eddy list file. This file should be in the SAMAS format. The header record contains the number of eddies. The remaining records contain the latitude, longitude, radius, type, and source code for the eddies. The

program does not use the source code.

Updating by day steps of -
The user is prompted for the number of days to progress the input Gulf Stream and eddies.

Final output North wall Gulf Stream file -
The user is prompted for the output North wall Gulf Stream file (upper). This file will be in the SAMAS format. The header record contains the number of positions. The remaining records contain the latitude, longitude, and source code for the Gulf Stream positions. The program will output a source code for these new positions of 'O'.

Final output South wall Gulf Stream file -
The user is prompted for the output South wall Gulf Stream file (lower). This file will be in the SAMAS format. The header record contains the number of positions. The remaining records contain the latitude, longitude, and source code for the Gulf Stream positions. The program will output a source code for these new positions of 'O'.

Final output Eddy file -
The user is prompted for the output eddy list file. This file should be in the SAMAS format. The header record contains the number of eddies. The remaining records contain the latitude, longitude, radius, type, and source code for the eddies. The program will output a source code for these new positions of 'ES'.

For more information on this type of edge detection see references 4 and 5.

2.6 Create Edge Image

The user can select this option from the menu as item 7, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]edge" will work just as well. The user may choose to run the functions used to create the edge image or to create a command file that can be used later to create the edge image. The edge function consists of 4 modules. This driver will provide the prompts required to execute all or some of the modules. Intermediate output files will be created and deleted for the user. The output file of 1 module will be the input for the next and only the last output image will be saved. The user may elect to run these modules separately and keep or delete the intermediate image files as desired. The user is prompted first for which of the modules are to be run. If the user selects to run cluster shade and any other module, then cluster test must also be run. The output from cluster shade can only be input to the function cluster test. The user may optionally clean, dilate, or thin the edge image. The clean and dilate are included in the same module, but either can be done without the other. The input for cluster shade is an IR image with an optional mask, perhaps for clouds. The input for cluster test is the output image from cluster shade. The inputs to the remaining modules can be any edge image in the SAMAS format. A prompt is given for a working directory. This is where the working files will be created, and if no error occurs they will be deleted. The user will be prompted for the parameters appropriate for the selected modules. The initial prompts for creating an edge image are as follows:

Process or create command file (def:Proc or Com) -
The user is prompted for the type of run. If Proc (process) is selected, then the programs selected by the user will be executed as soon as all the inputs have been entered. If Com (command) is selected, then the user will be prompted for a command file name and a command file will be created. This command file can be used for batching the creation of the edges.

Name for command file (def: Edge.Com) -
If the user has selected to create a command file, then this prompt is for the name of the command file. A default name has been provided.

Run Cluster Shade (def: YES or NO) -
The user is prompted for Yes or No for running the cluster shade module. This module creates an image of cluster shade values.

Run Cluster Test (def: YES or NO) -
The user is prompted for Yes or No for running the cluster test module. This module creates edges from a cluster shade output image.

Run Clean (def: YES or NO) -
The user is prompted for Yes or No for running the clean option of the Dilate/Clean module. Any edge image can be used as input to this module.

Run Dilate (def: YES or NO) -
The user is prompted for Yes or No for running the dilate option of the Dilate/Clean module. Any edge image can be used as input to this module.

Run Thin line (def: YES or NO) -
The user is prompted for Yes or No for running the thin line module. Any edge image can be used as input to this module.

Input image file -
The user is prompted for an image file to be used as input for the first module selected.

Output image file -

The user is prompted for an output image. This image will be the output of the last module selected.

Working directory for temporary files (def:current directory) -

The user is prompted for a directory for creating working files that are the output from one of the selected modules which is to be the input to the next selected module.

2.6.1 Cluster Shade

This algorithm computes an output image of cluster shade measures. These measures are derived from grey level co-occurrence (GLC) matrices computed on the overlapping windows of the input image. The (i,j)th element of the GLC matrix, $P(i,j|\Delta x,\Delta y)$, is the relative frequency with which two image elements, separated by distance $(\Delta x,\Delta y)$, occur in the image window, one with intensity level i and the other with intensity level j. Consider an M x N pixel image window with L intensity levels ranging from 0 to (L-1). Let f(m,n) denote the intensity level of pixel (m,n). Then

$$P(i,j|\Delta x,\Delta y) = \sum_{i=1}^{m-\Delta x} \sum_{j=1}^{n-\Delta y} A$$

where $A = 1 / (M-\Delta x \text{ X } N-\Delta y)$ if $f(m,n)=i$ and $f(m+\Delta x,n+\Delta y) = j$. Otherwise, A=0. $P(i,j|\Delta x,\Delta y)$ is therefore an L x L matrix of second order probabilities. In uniform areas of the image, two pixels displaced by $(\Delta x,\Delta y)$ are likely to have nearly equal intensities, i.e., i=j, which means that these pixels contribute to increased probabilities in the near-diagonal elements of the GLC matrix. By contrast, in areas of the image where the displacement $(\Delta x,\Delta y)$ spans an edge, i.e. i<<j or i>>j, the probabilities in the off-diagonal elements of the GLC matrix are increased accordingly. The cluster shade measure, $S(\Delta x,\Delta y)$ is computed for each GLC matrix as follows:

$$S(\Delta x,\Delta y) = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} (i+j-\mu_i-\mu_j)^3 \, P(i,j|\, \Delta x,\Delta y)$$

where,

$$\mu_i = \sum_{i=0}^{L-1} i \sum_{j=0}^{L-1} P(i,j|\ \Delta x, \Delta y)$$

$$\mu_j = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} [j\ P(i,j|\ \Delta x, \Delta y)]$$

The center point of each window of the image is replaced by the S(Δx,Δy) value computed from its neighbors, thus creating a 'cluster shade' image of S(Δx,Δy) values. Points that are on the output image edge, thus not the center of any window, are set to 0.

The user may chose not to compute GLC matrices, or cluster shade measures, for all windows of the input image. There are 2 methods allowed for eliminating pixels from the computations. If a mask input image is provided, then the nonzero values of this image will be used to mask pixels. No pixel of the original input image will be processed if there is a corresponding nonzero pixel in the mask input image. The second method is to eliminate pixels in the first input image that do not meet the threshold. If a positive threshold is specified, then no pixels with intensity greater than or equal to the threshold will be processed. If a negative threshold is specified, then no pixels with intensity less than or equal to the absolute value of the threshold will be processed. At least 75% of the pixels in a window must be considered good for a Cluster Shade value to be computed, else a 0 will be output for that window. The prompts for running this option are as follows:

Input mask image for Cluster Shade (opt.) -

The user is prompted for an optional input image to be used for masking. The algorithm does not process neighborhoods with less than 75% nonmasked values.

Window size x dimension (def: 16) -

The user is prompted for the number of samples in the Neighborhood or processing window to be used for computing the GLC matrix, which is used to compute the cluster shade measure.

Window size y dimension (def: 16) -

The user is prompted for the number of lines in the Neighborhood or processing window to be used for computing the GLC matrix which is used to compute the cluster shade measure.

Delta length x dimension (def: 0) -

The user is prompted for the number of samples in the displacement vector to be used for computing the GLC matrix which is used to compute the cluster shade measure.

Delta length y dimension (def: 0) -

The user is prompted for the number of lines in the displacement vector to be used for computing the GLC matrix which is used to compute the cluster shade measure.

Input processing threshold -

The user is prompted for an optional processing threshold. If the threshold is positive, then the algorithm does not use any values of the neighborhood, which are greater than or equal to this threshold. If the threshold is negative, then the algorithm does not use any values of the neighborhood, which are less than or equal to the absolute value of this threshold. The algorithm does not process neighborhoods with less than 75% nonmasked values.

2.6.2 Cluster Test

This function creates an edge image based on finding certain 0 crossings in the input cluster shade image. The first step is to find the 0 crossings, which satisfy the threshold specified as the initial threshold. Every pixel of the input cluster shade image is tested. If the absolute value of the pixel is greater than or equal to the initial threshold, then the 8 immediate neighbors of that pixel are tested. If any of the 8 immediate neighbors also has an absolute value greater than or equal to the initial threshold and is opposite in sign (+,-) from the center pixel of the 3 x 3 neighborhood, then an edge has been found. The center pixel of the 3 x 3 neighborhood is marked with a value of 255 in the output edge image. For no edge a value of 0 is output.

If the parameter for maximum passes has not been set to 0, then additional passes will be made through the image to attempt to extend some of the edges which were created by doing the initial 0 crossing test. If a pixel has not yet been marked as an edge and has a neighbor which has been marked as an edge in a previous pass, then that pixel is tested for a 0 crossing using the minimum threshold. If the pixel passes this 0 crossing test then, the pixel is marked as an edge pixel. After each pass, the number of newly added pixels is listed for the user. These passes are repeated until no new pixels are being added or a specified maximum passes have been made through the image to extend the edges. The output image is a byte image with values of 0 for the background and 255 for the edges. The prompts for this option are as follows:

Initial threshold (def: 20) -

This value is used to perform the initial 0 crossing test of the cluster shade measures. If a pixel has an absolute value greater than or equal to the initial threshold and any of the 8 immediate neighbors of that pixel also have an absolute value greater than or equal to the initial threshold and are opposite in sign from the center pixel of the 3 x 3 neighborhood, then a 0 crossing has been found.

Minimum threshold (def: 5) -

This value is used to perform 0 crossing tests for pixels that have not been marked as an edge but have any of its 8 immediate neighbors which have been marked as an edge in any previous pass through the image.

Maximum allowed passes (def: 30) -

After performing the initial 0 crossing test on the pixels of the input cluster shade image, the image will be processed again to extend the edges. The process of extending edges will be performed until no new edge pixels can be assigned or the process has been performed the maximum allowed passes after the first.

2.6.3 Dilate/Clean

This function performs 2 tasks, first to clean and then dilate (or thicken) an edge image. The user defines the size of the clean window. The clean window is moved over the entire input image. If a nonzero pixel or group of nonzero pixels is completely contained within the window with no nonzero pixels on the boundary of the window, then the window is cleaned. A window is cleaned by setting all pixels contained within the window to 0.

The second task of this function is dilation. This is done to thicken lines and in some cases as a result lines will be connected in the process. The user defines the number of times dilation will be performed on the input image. For each time of dilation all pixels are checked. If a pixel is nonzero, then that pixel and all 8 of its immediate neighbors will be set to 1. A line 1 pixel wide at the start will be 3 pixels wide after 1 step, 5 pixels wide after 2 steps, etc. The prompts for this option are as follows:

Window size x dimension (def: 16) -

Number of samples in the window to be used for image cleaning. The range for this parameter is 1 - 16.

Window size y dimension (def: 16) -

Number of lines in the window to be used for image cleaning. The range for this parameter is 1 - 16.

Number of dilate steps (def: 1)
Defines the number of iterations of dilation or line thickening to perform.

2.6.4 Thin Line

This function given a binary input image produces a skeleton of that image. All features of the original image are reduced to single pixel wide lines. Outer layers are peeled off, while endpoints and connectivity are preserved. See reference 6 for more information on the method used.

Given the following notations and definitions:

notation:

Q ... The set to be thinned
S ... The skeleton
B(Q) ... The contour of Q
L(Q) ... The set of pixels of B(Q) which are not multiple pixels
M(Q) ... The set of pixels of B(Q) which are multiple pixels
K(Q) ... The set of all pixels in B(Q) which have neighbors in S

4	3	2	D-neighbors are 1,3,5,7
5	P	1	I-neighbors are 2,4,6,8
6	7	8	

Definition 1. The contour of a set of pixels Q is defined as the set of pixels in Q which have at least 1 D-neighbor not in Q.

Definition 2. The C-neighbor of a pixel on a contour are defined as the previous and next pixels (possibly coinciding) found during contour tracing.

Definition 3. A pixel is said to be multiple if one or more of the following conditions hold:

(A) It is traversed more than once during tracing (i.e., at the completion of the process its value is greater than 2 if using actual tracing method).
(B) It has no neighbors in the interior of the region (i.e., no neighbor with value of 1).
(C) It has at least one D-neighbor, which belongs to the contour but is not one of its C-neighbors.

Definition 4. A skeletal pixel is one for which one of the following conditions is true:

(A) It is a multiple pixel.
(B) It has a D- or I-neighbor which has been identified as a skeletal pixel during an earlier tracing.
(C) (optional) Its 2 C-neighbors forms an angle of 90°.

The steps of the algorithm are as follows:

```
I.      Set S to the empty set
II.     while Q is not empty do steps III - VI.
        Begin
III.       Find B(Q), by contour tracing, use neighborhood check to
           compute B(Q) for all pixels of Q with a D-neighbor not an
           element of Q.
IV.        Find L(Q) and M(Q) by retracing B(Q) while checking for
           conditions of definition 3.
V.         Find K(Q) by examining all pixels in B(Q) for neighbors
           in S and, if desired also checking condition (C) of
           definition 4.
VI.        Set S = S U M(Q) U K(Q) and Q = Q - B(Q).
        END
VII.    End of algorithm.
```

2.7 Label Edge Image

The user can select this option from the menu as item 8, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]Label_Driver" will work just as well. The user may choose to run the functions used to create the labeled output or to create a command file that can be used later to create the labeled output. Feature labeling is done by a method of nonlinear probabilistic relaxation. The user supplies an IR image, an edge image created from the IR image, and a previous analysis for any or all of the following: North wall Gulf Stream, South wall Gulf Stream, and eddy list. The user may have run the previous analysis through the expert system prior to the labeling. This is done to progress the previous analysis to the time of the current image data. The function can accept at most 8 input objects. The North wall, if provided, is a single object. The South wall, if provided, is a single object. Each eddy of an eddy list file, if provided, is an object. The user must input at least one of the previous analysis: North wall, South wall, or eddies.

The user is given several output options, at least one of which should be chosen. The first option is for a labeled output image file. This image file will contain pixels with specific values corresponding to the labeled objects. The values correspond to bit planes of the output image. The values for each type of label will be listed for the user. The user may, however, choose to have the 3 types of objects output separate and a version of the edge image created less any or all of the labeled edges. The user is given the option of getting the North and South walls as list files. These list files will be a regular SAMAS Gulf Stream list file. The first record will contain the number of positions. The positions will be latitude, longitude, and a source code of 'O'. The points in this file are not consecutive points along a Gulf Stream. The points are random Gulf Stream positions. The user is prompted for a pixel distance for creating these list files. This is to reduce the amount of data being saved. No points within the specified distance to a point saved in the output list will be kept. The user can select to have just the eddy edges saved in an image file. Each eddy in the output file will have a different plane value associated with it. Finally, the user can have any of the labeled objects extracted from the input edge image to create an edge image with any or all of the North wall, South wall, or eddy edges missing. The prompts for creating the labeled output are as follows:

Process or create command file (def:Proc or Com) -

The user is prompted for the type of run. If Proc (process) is selected, then the labeling will be executed as soon as all the inputs have been entered. If Com (command) is selected, then the user will be prompted for a command file name and a command file will be created. This command file can be used for batching the creation of the labeled edges.

Name for command file (def: Label.Com) -

If the user has selected to create a command file, then this prompt is for the name of the command file. A default name has been provided.

Input grey scale image file -

The user must enter the IR image used to create the edge image which is to be labeled.

Input edge image file -

The user must enter the input edge image to be labeled.

Input North wall list file (opt.) -

The user can select to enter an optional input North wall list file for the previous analysis to be used to label the North wall edges of the input edge image. Prior to the labeling the user may have run this analysis through the expert system to match the input image time.

Input South wall list file (opt.) -

The user can select to enter an optional input South wall list file for the previous analysis to be used to label the South wall edges of the input edge image. Prior to the labeling the user may have run this analysis through the expert system to match the input image time.

Input eddy list file (opt.) -
The user can select to enter an optional input eddy list file for the previous analysis to be used to label the eddy edges of the input edge image. Prior to the labeling the user may have run this analysis through the expert system to match the input image time.

Output labeled edge image file (opt.) -
The user may select as output a labeled edge image. This image contains the edges that were labeled as North wall, South wall, and eddies. Each object North wall, South wall, and each eddy has a unique plane value associated with it. Labels can only be created if an input analysis for that object was provided. The plane values for the objects will be listed for the user. This is an optional output.

Output North wall list file (opt.) -
The user can select to have the North wall extracted from the labeled edge image and output as a list file. This is an optional output file and the prompt for it will be given only if an input North wall previous analysis was specified.

Pixel distance to separate Gulf Stream coding (def: 0) -
This prompt will be given only if an output North wall list file is specified. The pixel distance is used to reduce the number of positions output to the above specified file. Once a position has been selected for the list, then no positions within the specified distance will be kept for output.

Output South wall list file (opt.) -
The user can select to have the South wall extracted from the labeled edge image and output as a list file. This is an optional output file and the prompt for it will be given only if an input South wall previous analysis was specified.

Pixel distance to separate Gulf Stream coding (def: 0) -
This prompt will be given only if an output South wall list file is specified. The pixel distance is used to reduce the number of positions output to the file. Once a position has been selected for the list, then no positions within the specified distance will be kept for output.

Output eddy image file (opt.) -
The user can select to have the eddy edges extracted from the labeled edge image and output as an eddy image file. The image file will contain only the eddy edges, each labeled eddy will retain its unique plane value. This is an optional output file and the prompt for it will be given only if an input eddy previous analysis was specified.

Output frontal image file (opt.) -
The user can have any or all of the labeled edges masked from the input edge image to create a new frontal edge image.

Mask North wall from frontal image (Def: Yes) -
This prompt is given only if a frontal image file is to be produced and a North wall previous analysis was specified. This prompt determines if the edges labeled as North wall are to be masked from the original edge image.

Mask South wall from frontal image (Def: Yes) -
This prompt is given only if a frontal image file is to be produced and a South wall previous analysis was specified. This prompt determines if the edges labeled as South wall are to be masked from the original edge image.

Mask Eddies from frontal image (Def: Yes) -
This prompt is given only if a frontal image file is to be produced and an eddy previous analysis was specified. This prompt determines if the edges labeled as eddies are to be masked from the original edge image.

For more information on this type of labeling see reference 7.

2.8 Run CEOF - Gulf Stream Interpolation

The user can select this option from the menu as item 9, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]CEOF" will work just as well. The user may choose to run the function or to create a command file that can be used later for the same purpose. This function uses complex empirical orthogonal functions (CEOF) to create a continuous Gulf Stream North wall from fragmented segments and positions. The output Gulf Stream North wall from the labeling is usually used as the input to be interpolated. The input Gulf Stream can be entered as an image file along with the range of values for pixel intensities or planes required to extract the positions; or an actual Gulf Stream list file can be entered. If an input Gulf Stream list file is used as the input to be interpolated, then the code values associated with the positions are used to determine weighting. Positions with codes of 'A' for altimeter are weighted higher than positions with other code values. If an image file is used, then all positions are weighted equal. Also, input to this function is either a mode file containing the initial mode coefficients or a good Gulf Stream list file, one which gives a realistic description of a previous Gulf Stream. The user can provide a good quality previous analysis Gulf Stream as input to the INTERP option and a mode file will be created, or a mode file from a previous run of this function can be used. The user must also choose the number of eigenvectors to be used for the interpolation. Output from this function is a mode file and an interpolated continuous Gulf Stream created from the input partial Gulf Stream.

Process or create command file (def:Proc or Com) -

The user is prompted for the type of run. If Proc (process) is selected, then the function will be executed as soon as all the inputs have been entered. If Com (command) is selected, then the user will be prompted for a command file name and a command file will be created. This command file can be used for batching the creation of the interpolated Gulf Stream and associated mode file.

Name for command file (def: CEOF.Com) -

If the user has selected to create a command file, then this prompt is for the name of the command file. A default name has been provided.

Input labeled Gulf Stream -

The user must enter the Gulf Stream that is to be used for the interpolation. This Gulf Stream can be partial and need not be continuous. The file can be in the form of an image or a Gulf Stream list. If the file is an image file, then the user will be prompted for type of values, the range for the values, and a spacing increment.

Input file type (Def: List or Image) -

The user must define the type of input Gulf Stream file list or image. If the input file is image, then the user will need to answer the next 3 prompts, else they are skipped.

Type of extraction (Def: Values or Planes) -

The user must state whether the positions within the image file are intensity values or plane values. This information is used for extracting the Gulf Stream positions from the image.

Value limits for extraction (Def: 1 1) -
Plane limits for extraction (Def: 1 1) -

The user must supply the value or plane ranges minimum and maximum at one of these prompts for extracting the Gulf Stream positions from the input image file.

Minimum pixel distance between points (Def: 0) -

The user may choose to reduce the number of the position for the Gulf Stream extracted from the image file. No pixel position within this specified distance of a pixel already selected for the gulf steam list will be saved.

Run INTERP (Def:Yes or No) -

The user has the option of running INTERP on an input Gulf Stream list to create a mode file to be used to start the interpolation. If INTERP is run, then the user will be prompted for an input Gulf Stream file. If INTERP is not run, then the user will be prompted for an input mode file, which can be output from a previous run.

Input Gulf Stream for INTERP to create mode file -
The user must input a realistic previous Gulf Stream list file to be used to create a mode file for OPTO.

Input Mode file for Opto -
The user must enter a mode file to be used to start the interpolation process. This file could be the output of a previous run of this function.

Output Gulf Stream File -
The user must provide the name for the new Gulf Stream list file.

Output Mode File -
The user must provide the name for the mode file associated with the new Gulf Stream list.

Number of eigenvectors to be used for fitting (Def: 10) -
The user is prompted for the number of eigenvectors to be used for the interpolation of the Gulf Stream.

For more information on the CEOF interpolation see reference 8.

2.9 Run Hough - Eddy Detect

The user can select this option from the menu as item 10, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]Hough" will work just as well. The user may choose to run the function or to create a command file that can be used later for the same purpose. Input to this function is a labeled edge image. The image may contain eddies only or be of mixed labels. If the image is of mixed labels, then the user will be required to supply the type of values (pixel intensity or planes) and a range for extracting just the eddy edges. The user is also allowed an option for dilating the edges prior to the eddy detection.

Given a single radius the Hough Transform is applied as follows. An accumulator array is created. This array will be the same size as the image array and will be initially set to zeros. The image array is scanned for edge pixels. When an edge pixel is found, then all entries in the accumulator array are incremented which are the radius distance from that pixel's corresponding accumulator coordinates. These entries represent the center coordinates of all circles of the given radius that would contain on their boundary that edge pixel.

The user may have selected a width greater than 1. In this case the Hough Transform will be applied for a donut or set of concentric circles. The minimum and maximum radius values are converted from kilometers to pixels. For each radius from minimum radius to maximum radius, incrementing by the radius increment, the accumulator array will be set to 0, the Hough Transform applied for that radius or set of radii; the entries selected from the accumulator array, which meet the threshold test will be stored in a list. The list will contain the radius, center coordinates, and normalized accumulator value. When width is greater than 1 the Hough Transform is applied multiple times without resetting the accumulator array to zeros. Width defines the number of radii for which the Hough Transform will be applied without resetting the accumulator array to zeros. The user is given a parameter for type of weighting. This parameter is used to define weights for the radii. If type of weighting is 'Ones', then all radii are weighted equal. If type of weighting is 'Decrease', then the smaller radius is weighted by 1 incrementing by 1 up to the largest radius of the donut. If type of weighting is 'Increase', then the larger radius is weighted by 1 incrementing by 1 to the smallest radius of the donut. The radius that will be stored in the list will be the larger radius of the donut. The accumulator entries are normalized so that they can be tested against a percentage threshold. Given the number of pixels for each of the radius and the desired weighting for each of the radius a maximum possible accumulator value is computed. The computed accumulator values are divided by this maximum possible value. This normalized value is then compared with the percentage threshold that was specified. The entries, which are equal or greater than the threshold, are stored in a list for further processing.

After the Hough Transform has been applied for all the desired cases, the list of entries, which passed the threshold test, is sorted. The entries are sorted first in descending order by the radius and second in descending order by the normalized accumulator values. The function is biased to the larger eddies. Each entry is checked against the preceding entries in the list. A circle will be eliminated from the list if it is found to overlap another circle defined in a previous entry in the list.

The edited list of center image coordinates, pixel radius, and normalized accumulator values must be converted to a list of output eddies. The image sample, line coordinates are converted to latitude, longitude coordinates. The pixel radius is converted to kilometers. The center latitude, center longitude, kilometer radius, type code (U for Undefined) and source code (ED) are all written to the output eddy list file.

The prompts for eddy detection are as follows:

Process or create command file (def:Proc or Com) -

The user is prompted for the type of run. If Proc (process) is selected, then the function will be executed as soon as all the inputs have been entered. If Com (command) is selected, then the user will be prompted for a command file name and a command file will be created. This command file can be used for batching the eddy detection.

Name for command file (def: Hough.Com) -

If the user has selected to create a command file, then this prompt is for the name of the command file. A default name has been provided.

Input edge image -

The user must input the labeled image containing the eddy edges. This image may contain eddy edges only or contain mixed edges. The user must state which case applies in the next prompt.

Input file type (Def: Eddies or Mixed) -

The user must state whether the eddy image file contains only eddy edges or contains other types of edges also. If the image is of mixed labels, then the user must answer the next 2 prompts to define the edges to be considered as eddy edges.

Type of extraction (Def: Values or Planes) -

The user must specify whether the eddy edges are to be extracted by intensity value or plane values. This is used for mixed label input images only.

Value limits for extraction (Def: 1 1) -
Plane limits for extraction (Def: 1 1) -

The user must define the data ranges for extracting the eddy labels from the input mixed label image. The prompt will be for intensity values or planes depending on the previous parameter.

Dilate eddy edges (Def: Yes)? -

The user can choose to have the input eddy edges dilated. Dilated edges usually work better for eddy detection, since eddies are not truly circular.

Number of desired dilate steps (def: 1) -

If the edges are to be dilated, then the user must specify how much dilation is desired. For a single pass of dilation the 8 neighbors of any edge pixel also are set to be edge pixels. Single pixel edges will become 3 pixels wide for a single pass of dilation.

Output eddy list file -

The user must provide the output eddy file name. If no eddies are found, then no file will be created and a message will be given.

Minimum and maximum radius to consider (Km) (Def: 50.000000 133.000000) -

The user must specify the minimum and maximum radius values to be processed. This value is in kilometers.

Circle mask pixel width (Def: 1) -

The user must enter the width or number of pixel radii to be used in detecting circles for each increment of radius.

Radius check pixel increment (Def: 2) -

The user may select not to test for all radii between the selected minimum and maximum radius values. An increment of 2 means every other radius will be tested.

Type of weighting ones, decrease to center, or increase to center (Def:O*nes, D*ecrease, I*ncrease) -

This parameter is only meaningful if the width value is greater than 1. When using multiple radii for detecting circles the concentric circles forming the donut can all be weighted equal 'Ones', the outer radii can be weighted higher than the inner radii 'Decrease', or the output radii can be weighted less than the inner radii 'Increase'. The range of weights for the decrease and increase is 1 incrementing by 1 for each new radii of the donut.

Threshold for circles (percentage 0 - 100) (Def: 40.000000) -

This threshold is used to define which circles will be kept in the list for sorting. No circles with a normalized accumulator value less than this threshold will be considered. The weights are normalized by taking the weighted sum of points of the donut or single radius and dividing by the maximum possible sum.

2.10 Apply Image Mask

The user can select this option from the menu as item 11, or the command "Run Vax$User1:[Peckinpaugh.SAMAS.Exe]Mask" will work just as well. This function is used to apply a mask to an image file. The user enters the image to be masked and 1 or more images to be used as masks. The user selects the option of masking the zero or nonzero values of the mask images. If the zero values of the mask images are to be used, then for all values of the input mask images which are zero the constant value specified by the user will be used as the output value, else the value from the background image will be used. If the nonzero values of the mask images are to be used, then for each value of the input mask images that are nonzero the constant value specified by the user will be used as the output value, else the value from the background image will be used. The prompts for this function are as follows:

Background image -
The user must specify the background image. The nonmasked values of the output image will have the background image values.

Mask image -
This prompt will be given multiple times until no name is entered. The user can enter 1 or more images to be used as masks. Either the 0 or nonzero values of these images will be set to a constant value in the output image.

1 - Mask with constant for non-zero values of mask
2 - Mask with constant for zero values of mask
Select option from list (Def: 1) -
The user must select whether the zero or nonzero values of the input mask images are to be used.

Input constant value to be used (Def: 0) -
The user must select the constant value to be used for the masked pixels of the output image.

Output image -
The user must supply the name for the output image file. The data type and mapping information will be the same as for the input background image.

2.11 Vax Command

The user can select this option from the menu as item 12. A prompt will be given and the user can enter a regular Vax command. This allows the user to do such a thing as batch a command file, list a directory, etc. Some Vax commands do not work in this mode, example: Assign.

3. SAMAS Source and Data Files

The SAMAS system is a collection of software modules bound together by a driver program. The driver program uses a data file to define the modules that will be part of the system. This driver can be used for any collection of independent modules provided the data file exists to define them. The user is told in section 2 how to run the driver program using the defaults. The driver can accept 2 command line parameters. The first parameter defines the data file that contains both the module descriptions for the menu and the file names for the executables. The second parameter defines the title to be used for the menu. For the VAX to accept command line arguments a symbol must be defined. To define the symbol 'SAMAS' for the driver executable the following command is used:

```
SAMAS == "$ VAX$User1:[Peckinpaugh.SAMAS.Screen]Driver.Exe"
```

Using this symbol the user can execute the SAMAS driver with or without parameters. The user can use the Run command and the executable file name to run the driver with the default parameters.

The user may choose to use a list of commands other than those in the default list file. This list of commands need not even be for SAMAS. The first parameter to this function is the data file name. The data file defines the menu selections. The format for this file is 2 records for each entry in the menu list, not counting the last option - Quit. The file is ASCII and can be generated using the general purpose editor. The first record is a description for the entry. This along with an entry number assigned by the program is what will appear when the menu is displayed. The second record contains the full name of the file that will be executed for that menu entry. An example record pair for item 3 of the default SAMAS menu is as follows:

```
Dump Image Values
Vax$User1:[Peckinpaugh.SAMAS.Exe]Dump_Image.Exe
```

No quotes are required and no blank lines are allowed preceding or following the record pair. New modules can easily be added to SAMAS in the future by simply adding entries to the command list file.

The second parameter can be used to define a title for the menu. The title can be one word or a group of words in double quotes. The default title for example is "Semi-Automated Mesoscale Analysis System".

The source for the driver program is in the following file:

```
     Vax$User1:[Peckinpaugh.SAMAS.Screen]Driver.C
```

The default command list file is in the following file:

```
     Vax$User1:[Peckinpaugh.SAMAS.Screen]SAMAS_Command.Lis
```

The following commands are required to compile and link the driver:

```
     define Lnk$Library sys$library:vaxccurse.olb
     define Lnk$Library sys$library:vaxcrtlg.olb
     define Lnk$Library sys$library:vaxcrtl.olb
     CC Vax$User1:[Peckinpaugh.SAMAS.Screen]Driver
     Link Driver
```

A command file has been created to build the default SAMAS system. This command file will compile and link all modules for all functions and drivers. The command file is as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Comp_Link.Com
```

See appendix IV for a listing of Comp_Link.Com.

3.1 SAMAS Standard Input/Output Files

The standard input and output files for the SAMAS are Gulf Stream list, eddy list, and image. These files have defined formats. All modules developed for the system will use these formats for these types of files.

3.1.1 Gulf Stream List File Format

The Gulf Stream list files are ASCII. The first record contains the number of positions contained in the file. The remaining records contain the Gulf Stream positions--latitude, longitude, and a source code. The source code values that have been established thus far are 'I' for image, 'A' for altimeter, and 'O' for other. An example of Other would be CEOF interpolated positions. Consecutive positions with code values of 'I' are expected to be consecutive connectable points of the Gulf Stream list. The format for these records is as follows:

Header (Record 1)		
I5	Number of positions contained in the file	
Positions (Record 2 - EOF)		
F10.4	Latitude	(+ North, - South)
F10.4	Longitude	(+ East, - West)
A2	Source Code	(I image, A altimeter, O other)

See appendix I for C and FORTRAN examples of reading and writing Gulf Stream list files.

3.1.2 Eddy List File Format

The eddy list files are ASCII. The first record contains the number of eddies contained in the file. The remaining records contain the eddy information, one eddy per record. The eddy information consists of center latitude, center longitude, kilometer radius, eddy type (Warm, Cold, or Undefined), and a source code. The source code is for the analyst to trace the origin of an eddy. Some functions, such as the Expert System assign source codes for output eddies. The source code output by the Expert System is 'ES'. The format for these records is as follows:

Header (Record 1)

I5	Number of eddies contained in the file

Eddy (Record 2 - EOF)

F10.4	Center Latitude	(+ North, - South)
F10.4	Center Longitude	(+ East, - West)
F10.4	Radius	(Kilometers)
A2	Eddy Type	(W warm, C cold, U undefined)
A3	Source Code	(2 character code)

See appendix II for C and FORTRAN examples of reading and writing Eddy List files.

3.1.3 Image File Format

The image data files have an ASCII header record and binary data records for the image data. The header record contains the information number of samples per line of data, number of lines of data, data type, and map registration information. The map registration information consists of North latitude limit, West longitude limit, degree/pixel ratio, and projection type. The format for the image file records is as follows:

Header (Record 1; ASCII)

I5	Number of samples of data per line (record)
I5	Number of lines of image data (records)
I5	Data type (1 Byte, 2 Integer*2, 3 Integer*4, 4 Real)
F10.4	North latitude limit for map registration (+ North, - West)
F10.4	West longitude limit for map registration (+ East, - West)
F10.4	Degree/pixel ratio for map registration
I5	Projection type for map registration (0 Rectilinear, 1 Mercator)

Data (Record 2 - EOF; Binary)

The length of the data record depends on the data type. For Byte data there is 1 byte per sample. For integer*2 there is 2 bytes per sample. For integer*4 and real there is 4 bytes per sample. The last 2 bytes of the record are for end of record characters. The record length thus is the number of samples times the number of bytes required for the data type plus 2 bytes.

For a description of the source code used to read and write data to the image files see section 3.2.2 of this document.

3.2 SAMAS Common Source

There is some source code that has been developed for use by many of the SAMAS functions. These are for allocating and releasing memory, image I/O, and map registration of image data.

Several of the functions listed in the SAMAS menu require multiple executable files. For these functions driver programs have been created. These driver programs create command files that execute the modules after getting the user input or in some cases, where specified by the user, can be used to create command files for later use. Code has been developed for these driver programs to get user input values and for the command files remove (delete) working files created by the program modules.

3.2.1 Memory Allocation and Release

The C language provides means for dynamic memory allocation and deallocation via such intrinsics as malloc and free. Code is provided for FORTRAN modules of SAMAS to also get dynamic memory allocation and deallocation. The source consists of both FORTRAN and C. The following files contain that source:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_F.For
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_C.C
```

To allocate memory for a FORTRAN module use the following call:

```
Idx = fGetMem( Buffer, Sz, Nvals )
```

Idx	Integer*4	Returned	Index used in Buffer to access the first element of the allocated memory
Buffer	?(*)	Passed	The Buffer to be used for accessing the allocated memory (User defines type)
Sz	Integer*4	Passed	The size in bytes of the elements of Buffer
Nvals	Integer*4	Passed	The number of Sz size elements required

To free the memory allocated by the call above for a FORTRAN module use the following call:

```
     fFreeMem( Buffer(Idx))
```

An example of the use for these modules is as follows:

```
     Integer Buffer(1), fGetMem
     Idx = fGetMem( Buffer, 4, 512 )

     i = Idx
     Do k = 1, 512
        Buffer(i) = k
        i = i + 1
     EndDo

     Call FillBuff( Buffer(Idx) )

     Call fFreeMem( Buffer(Idx))
     Stop
     End

     Subroutine FillBuff( Buffer )
     Integer Buffer(*)
     Do k = 1, 512
        Buffer(k) = k
     EndDo
     Return
     End
```

3.2.2 Image I/O

The source code used to read and write the image file data is found in the following file:

```
     Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For
```

To use the standard image I/O routine the user calls the FORTRAN subroutine ImageIO with the appropriate parameters. This routine is used to open, close, read, and write data of an image file. A description of the parameters is as follows:

```
     Image_IO ( File_Number, Mode, IO_Error, Buffer, Buffer_Type,
          Line_Number, File_Type, Number_Samples, Number_Lines,
          North, West, Degree_Pixel, Projection )
```

File_Number   FORTRAN unit number associated with the opened file

Mode	Defines the type of operation to be performed by ImageIO 1 - Open file and read header information 2 - Open file and write header information 3 - Read the specified line of image data 4 - Read the next line of image data 5 - Write the next line of image data 6 - Close the image file
IO_Error	IOstat value returned from the FORTRAN open, read, write, or close
Buffer	Contains the image file name for open and contains a line of data for read and write operations
Buffer_Type	Defines the type of data in the buffer being sent for a write or returned by a read (1 Byte, 2 Integer*2, 3 Integer*4, 4 Real)
Line_Number	Used when reading a specific line of data from the image file
File_Type	Defines the type of image data in the file (1 Byte, 2 Integer*2, 3 Integer*4, 4 Real)
Number_Samples	Number of samples contained in a line of image data
Number_Lines	Number of lines of image data in the file
North	North limit for the image map registration
West	West limit for the image map registration
Degree_Pixel	Degree/Pixel ratio for the image map registration
Projection	Type of map projection (0 rectilinear, 1 mercator)

The table below defines how to use the arguments for the different mode values:

ImageIO Mode Values and Arguments

Mode → Value Arguments ↓	1 Open Read	2 Open Write	3 Read Select Record	4 Read Next Record	5 Write Next Record	6 Close
File_Number	Return Integer	Return Integer	Pass Integer	Pass Integer	Pass Integer	Pass Integer
Mode	Pass Integer	Pass Integer	Pass Integer	Pass Integer	Pass Integer	Pass Integer
IO_Error	Return Integer	Return Integer	Return Integer	Return Integer	Return Integer	Return Integer
Buffer	Pass Integer	Pass Integer	Return *	Return *	Pass *	Not Used
Buffer_Type	Not Used	Not Used	Pass Integer	Pass Integer	Pass Integer	Not Used
Line_Number	Not Used	Not Used	Pass Integer	Not Used	Not Used	Not Used
File_Type	Return Integer	Pass Integer	Not Used	Not Used	Not Used	Not Used
Number_Samples	Return Integer	Pass Integer	Not Used	Not Used	Not Used	Not Used
Number_Lines	Return Integer	Pass Integer	Not Used	Not Used	Not Used	Not Used
North	Return Real	Pass Real	Not Used	Not Used	Not Used	Not Used
West	Return Real	Pass Real	Not Used	Not Used	Not Used	Not Used
Degree_Pixel	Return Real	Pass Real	Not Used	Not Used	Not Used	Not Used
Projection	Return Integer	Pass Integer	Not Used	Not Used	Not Used	Not Used

For examples of how to call the ImageIO routine from both C and FORTRAN see appendix III.

3.2.3 Map Registration

Each image data file contains as part of the header record information defining the map registration for that image. These parameters include the North limit, West limit, degree/pixel ratio, and map projection type (rectilinear or mercator). These values are written to the image header when the file is opened for write. Code is provided to do the following convert sample, line positions to latitude, longitude positions; convert latitude, longitude positions to sample, line positions; convert pixel distance to kilometer distance; convert kilometer distance to pixel distance; and compute the kilometer distance between 2 latitude, longitude positions. The source code is located in the following file:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_LL.For
```

To use the subroutines for computing latitude, longitude, sample, line, etc., the map array must first be initialized. This can be done in 1 of 2 ways. The SAMAS image name can be provided and the initialize routine will open the image file, read the header, and close the file using ImageIO described in the previous section. The second way is to supply the routine with the map information values. The call to initialize the map array given a file name is as follows:

```
Map_Init( fName, Relative, Map_array )
```

fName	Integer*4(*)	Passed	ASCII image file name
Relative	Integer*4	Passed	Sample, line values will be 0 relative if set to 0 or 1 relative if set to 1
Map_Array	Real(9)	Returned	Mapping array

The call to initialize the map array given the mapping information is as follows:

```
Map_Init( '20202020'X, Relative, Map_Array, North, West,
     DegPixel, Projection, Nline )
```

'20202020'X	Constant	Passed	4 blanks to signal no file name provided
Relative	Integer*4	Passed	Sample, line values will be 0 relative if set to 0 or 1 relative if set to 1
Map_Array	Real(9)	Returned	Mapping array

North	Real	Passed	North limit of the image
West	Real	Passed	West limit of the image
DegPixel	Real	Passed	The degree/pixel ratio for the image
Projection	Integer*4	Passed	Type of map projection (0 rectilinear, 1 mercator)
Nline	Integer*4	Passed	The number of lines in the image

Once the mapping array has been initialized, then it can be used for computing the coordinate conversions and distances. The following call computes the latitude position from a Y position:

```
Comp_Lat_From_Y( Y, Lat, Map_Array )
```

Y	Integer*4	Passed	Y is the line position, 0 or 1 relative depending on the value of Relative in the initialization of Map_Array
Lat	Real	Returned	The computed latitude value
Map_Array	Real(9)	Passed	Map array returned from Map_Init

The following call computes the longitude position from an X position:

```
Comp_Lon_From_X( X, Lon, Map_Array )
```

X	Integer*4	Passed	X is the sample position, 0 or 1 relative depending on the value of Relative in the initialization of Map_Array
Lon	Real	Returned	The computed longitude value
Map_Array	Real(9)	Passed	Map array returned from Map_Init

The following call computes the Y position from a latitude position:

```
Comp_Y_From_Lat( Lat, Y, Map_Array )
```

Lat	Real	Passed	The latitude position

Y	Integer*4	Returned	Computed Y or line position is 0 or 1 relative depending on the value of Relative in the initialization of Map_Array
Map_Array	Real(9)	Passed	Map array returned from Map_Init

The following call computes the X position from a longitude position:

```
Comp_X_From_Lon( Lon, X, Map_Array )
```

Lon	Real	Passed	The longitude position
X	Integer*4	Returned	Computed X or sample position is 0 or 1 relative depending on the value of Relative in the initialization of Map_Array
Map_Array	Real(9)	Passed	Map array returned from Map_Init

Images are mapped using a degree/pixel ratio. The relationship of kilometers to degrees varies with latitude and differs for latitude and longitude. A routine has been developed to convert a pixel distance to kilometer distance. The routine computes Kilometer distance from the given x, y position South the specified pixel distance and from the given X, Y position east the specified pixel distance. These 2 distances are averaged to get the returned kilometer distance. The following call converts the pixel distance at the specified X, Y position to kilometers:

```
Pixels_to_Km( X, Y, Pdis, KMdis, Map_Array )
```

X	Integer*4	Passed	X position for start of pixel distance
Y	Integer*4	Passed	Y position for start of pixel distance
Pdis	Integer*4	Passed	Pixel distance to be converted to kilometers
KMdis	Real	Returned	Computed kilometer distance for the pixel distance
Map_Array	Real(9)	Passed	Map array returned by Map_Init

The conversion of a kilometer distance to a pixel distance is done by the same principle as that described above. Now the position is specified as latitude, longitude and the conversion is from kilometers to pixels. The following call computes the kilometer distance centered at the specified latitude, longitude position to pixels:

```
KM_to_Pixels( Lat, Lon, KMdis, Pdis, Map_Array )
```

Lat	Real	Passed	Latitude position for start of kilometer distance
Lon	Real	Passed	Longitude position for start of kilometer distance
KMdis	Real	Passed	Kilometer distance to be converted to pixel distance
Pdis	Integer*4	Returned	Pixel distance computed for the kilometers distance
Map_Array	Real(9)	Passed	Map array returned by Map_Init

The last routine provided for mapping computes a kilometers distance between two latitude, longitude positions. This is not image dependent and thus does not require that Map_Init be run. The following call will compute a kilometer distance between two latitude, longitude positions:

```
KM_Dist( Dis, Lon1, Lon2, Lat1, Lat2 )
```

Dis	Real Returned	Computed distance in kilometers
Lon1	Real Passed	Longitude for the first position
Lon2	Real Passed	Longitude for the second position
Lat1	Real Passed	Latitude for the first position
Lat2	Real Passed	Latitude for the second position

3.2.4 Driver Input from User

Code is provided for getting real, integer*4, and character data from the user. The source for this code is in the following file:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Input.C
```

There are 3 subroutines one for each of the 3 types of data--real, integer, and character. Each routine will allow the user to hit RETURN to indicate that the defaults or no input is desired. The call for the function to input character data is as follows:

```
void InputChar( Input, Prompt )
```

Input	char[]	Returned	Contains string input by the user, if no string is entered, then no change is made to the buffer
Prompt	char[]	Passes	String used to prompt the user for input

The call for the function to input float or real data is as follows:

```
void InputFloat( Prompt, Nval, Val, DefVal )
```

Prompt	char[]	Passed	String used to prompt the user for input
Nval	int*	Returned	Number of values entered by the user, if no values are entered by the user, then this will be set equal to DefVal
Val	float[]	Returned	Values entered by the user, if no values are entered, then no change is made to this argument
DefVal	int	Passed	Value to be assigned to Nval if no values are entered by the user, should denote the number of default values contained in Val

The call for the function to input integer data is as follows:

```
void InputInt( Prompt, Nval, Val, DefVal )
```

Prompt	char[]	Passed	String used to prompt the user for input

Nval	int*	Returned	Number of values entered by the user, if no values are entered by the user, then this will be set equal to DefVal
Val	int[]	Returned	Values entered by the user, if no values are entered, then no change is made to this argument
DefVal	int	Passed	Value to be assigned to Nval if no values are entered by the user, should denote the number of default values contained in Val

3.2.5 Remove File

An executable module has been provided to remove (or delete) files. This is used by some of the drivers to delete intermediate files created by one module of a function to be used by another module of the same function. The source for this module is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Remove.C
```

To use the executable as a VAX type command it must first be defined. This can be done as follows:

```
Remove == "$ Vax$User1:[Peckinpaugh.SAMAS.Exe]Remove.Exe"
```

The command can then be used to delete a file as follows:

```
Remove fname
```

Where fname is the file to be deleted.

3.3 User DE and BE

The first 2 options of the SAMAS menu are for an IIS user DE and BE. These versions of DE and BE have a very limited set of commands. For a list of the commands see section 2.1. The functions that are unique to the SAMAS user DE and BE are as follows:

```
     cpu'samas'image_display
     cpu'samas'image_format
     cpu'samas'image_graphics
     m75'eddy'edit
     m75'frontal'edit
     m75'gulf_stream'edit
```

These user versions of De and Be can be executed outside of the menu by running the following executable files:

```
     Vax$User1:[Peckinpaugh.SAMAS.Exe]IIS_De.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]IIS_Be.Exe
```

Or, the following commands can be used:

```
     UserDe Vax$User:[Peckinpaugh.SAMAS.Exe]
     UserBe Vax$User:[Peckinpaugh.SAMAS.Exe]
```

The following executable files are required for either method of running the SAMAS user De and Be:

```
     Vax$User1:[Peckinpaugh.SAMAS.Exe]De.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Be.Exe
```

To create these user De and Be executables the following commands are used:

```
     For Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For
     For Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_LL.For
     For Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_F.For
     CC  Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_C.C
     For Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Display.For
     For Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Format.For
     For Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Graphics.For
     For Vax$User1:[Peckinpaugh.SAMAS.Src]EddyEdit.For
     For Vax$User1:[Peckinpaugh.SAMAS.Src]GS_Edit.For
     For Vax$User1:[Peckinpaugh.SAMAS.Src]FrontEdit.For
     CDC Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Display.CDC
     CDC Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Format.CDC
     CDC Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Graphics.CDC
     CDC Vax$User1:[Peckinpaugh.SAMAS.Src]EddyEdit.CDC
     CDC Vax$User1:[Peckinpaugh.SAMAS.Src]GS_Edit.CDC
     CDC Vax$User1:[Peckinpaugh.SAMAS.Src]FrontEdit.CDC
     BuildExec/Obj=( SAMAS_Display, SAMAS_Format, SAMAS_Graphics,
          EddyEdit, GS_Edit, FrontEdit, ImageIO, Map_XY_LL,
          Memory_F, Memory_C ) De, Be
```

For more information on the UserDe, UserBe, CDC and BuildExec commands see reference 9.

The source code for using the Menu, IIS_De, or IIS_Be for running the user De and Be require the additional source files listed below:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]IIS_De.C
Vax$User1:[Peckinpaugh.SAMAS.Src]IIS_Be.C
```

IIS requires a command definition file (.CDC) for each function in the UserDe and UserBe. This file contains the information required to run the function and get the input from the user. For a description of what is required in this file see reference 9.

The functions listed above all use IIS intrinsics for IIS image I/O, user input, screen manipulation, etc. For more information on IIS intrinsics see reference 10.

3.3.1 CPU'SAMAS'Image_Display

The source files, including the command definition file, for this function are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Display.For
Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Display.CDC
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For          3.2.2
```

The output for this function is to the IIS image display screen.

3.3.2 CPU'SAMAS'Image_Format

The source files, including the command definition file, for this function are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Format.For
Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Format.CDC
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_F.For          3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_C.C            3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For          3.2.2
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_11.For         3.2.3
```

3.3.3 CPU'SAMAS'Image_Graphics

The source files, including the command definition file, for this function are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Graphics.For
Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Graphics.CDC
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For         3.2.2
```

The output for this function is to the IIS image display screen.

3.3.4 M75'Eddy'Edit

The source file, including the command definition file, for this function are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]EddyEdit.For
Vax$User1:[Peckinpaugh.SAMAS.Src]EddyEdit.CDC
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_F.For         3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_C.C           3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For         3.2.2
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_11.For        3.2.3
```

3.3.5 M75'Frontal'Edit

The source file, including the command definition file, for this function are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]FrontEdit.For
Vax$User1:[Peckinpaugh.SAMAS.Src]FrontEdit.CDC
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_F.For         3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_C.C           3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For         3.2.2
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_11.For        3.2.3
```

3.3.6 M75'Gulf_Stream'Edit

The source file, including the command definition file, for this function are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]GS_Edit.For
Vax$User1:[Peckinpaugh.SAMAS.Src]GS_Edit.CDC
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_F.For          3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_C.C            3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For          3.2.2
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_11.For         3.2.3
```

3.4 Dump Image Values

This function is the third option of the SAMAS menu. The source required for this function is in the following files:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Dump_Image.C
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For          3.2.2
```

The executable file for this function is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Dump_Image.Exe
```

This file can be used with the Run command to initiate the function with out the menu driver. Output from this function is printed to the user's default output list device, usually the terminal screen.

3.5 Enter List - Eddy

This function is the fourth option of the SAMAS menu. The source required for this function is in the following files:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Eddy_EnterList.For
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_11.For         3.2.3
```

The executable file for this function is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Eddy_EnterList.Exe
```

This file can be used with the Run command to initiate the function with out the menu driver.

3.6 Enter List - Gulf Stream

This function is the fifth option of the SAMAS menu. The source for this function is in the following file:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]GS_EnterList.For
```

The executable file for this function is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]GS_EnterList.Exe
```

This file can be used with the Run command to initiate the function with out the menu driver.

3.7 Expert System

This function is the sixth option of the SAMAS menu. The source code for this function is written in OPS83, C, and FORTRAN. The Source for this function is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]GRAPHRTN.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]IO.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]MATHE.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]MOVEGS.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]NRMLZ.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]CCRULES.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]EDDIES.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]EDDYTYPES.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]FINALOUTPUT.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]RACYCLE.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]REGIONS.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]SETUP.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]WCRULES.OPS
Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]FO.FOR
```

The executable file for this function is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Expert.Exe
```

A command file has been created to compile and link the source to create this function. The command file is as follows:

```
Vax$user1:[Peckinpaugh.SAMAS.Src.Expert]CXlink.Com
```

See appendix IV for a listing of this command file.

Data files not specified by the user are required for this function. These required files include the following:

```
Vax$User1:[Peckinpaugh.SAMAS.Dat]Regions.Dat
Vax$User1:[Peckinpaugh.SAMAS.Dat]Movegs.Dat
Vax$User1:[Peckinpaugh.SAMAS.Dat]NomGS.Lat
Vax$User1:[Peckinpaugh.SAMAS.Dat]NomGS.Long
```

In the user's working directory any of the following working files may appear:

```
Expert.Tmp
NomGS.Lat
NomGS.Long
```

These files can be deleted by the user.

For more information on how the expert system actually works see reference 3.

3.8 Create Edge Image

This function is the seventh option of the SAMAS menu. This function uses a driver program. This function can be run interactively or a command file can be created by the driver to run later. There are 5 executable modules that can be used by the created command file. The command file created by the driver manages the deletion of the working or intermediate files created by the modules. The driver allows the user to select which modules of the edge creation are to be executed. The source for the driver is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Edge.C
Vax$User1:[Peckinpaugh.SAMAS.Src]Input.C                 3.2.4
```

The executable file for the driver program is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Edge.Exe
```

The source for the executable modules, which can be included in the command file, are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Cluster_Shade.For
Vax$User1:[Peckinpaugh.SAMAS.Src]Cluster_Test.For
Vax$User1:[Peckinpaugh.SAMAS.Src]Dilate_Clean.For
Vax$User1:[Peckinpaugh.SAMAS.Src]Thin_Line.For
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_F.For            3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Memory_C.C              3.2.1
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For            3.2.2
Vax$User1:[Peckinpaugh.SAMAS.Src]Remove.C                3.2.5
```

The driver program creates a command file that can either be run after getting all required inputs from the user, or if the user selects, the command file is created and saved for later use. The executable modules that can be used in the command file are any or all of the following:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Cluster_Shade.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]Cluster_Test.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]Dilate_Clean.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]Thin_Line.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]Remove.Exe                 3.2.5
```

These modules create the cluster shade image, threshold the cluster shade image to create an edge image, dilate and clean an edge image, thin the edge image, and delete files.

The user may have appear in his working directory any of the following working files:

```
CS000.Tmp
CT000.Tmp
DC000.Tmp
```

These files should be deleted by the command file, unless an error occurs. If these files do exist, then they can be deleted.

See references 4 and 5 for more information.

3.9 Label Edge Image

This function is the eighth option of the SAMAS menu. This function uses a driver program. This function can be run interactively or a command file can be created by the driver to run later. All modules of the labeling function are required and will be run, the user does not select which modules are executed. The command file created by the driver manages the deletion of the working or intermediate files created by the modules. The source for the driver is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src.Label]Label_Driver.C
Vax$User1:[Peckinpaugh.SAMAS.Src]Input.C                    3.2.4
```

The executable file for the driver program is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Label_Driver.Exe
```

The source for the executable modules, which are included in the command file created by this driver, are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src.Label]Rdata.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Label]A1.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Label]A2.C
Vax$User1:[Peckinpaugh.SAMAS.Src.Label]Label.C
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For          3.2.2
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_LL.For         3.2.3
Vax$User1:[Peckinpaugh.SAMAS.Src]Remove.C              3.2.5
```

The driver program creates a command file that can either be run after getting all required inputs from the user, or if the user selects, the command file is created and saved for later use. The executable modules which are used in the command file are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Rda_a.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]A1.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]A2.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]Label.Exe
Vax$User1:[Peckinpaugh.SAMAS.Exe]Remove.Exe            3.2.5
```

Data files not specified by the user are required for this function. These required files include the following:

```
Vax$User1:[Peckinpaugh.SAMAS.Dat]Angle.Dat
Vax$User1:[Peckinpaugh.SAMAS.Dat]Dist.Dat
```

The user may have appear in his working directory any of the following working files:

```
Apr.Image
Mean.Image
Label111.Tmp
Label222.Tmp
obj000.Tmp
```

These files should be deleted by the command file, unless an error occurs. If these files do exist they can be deleted.

See reference 7 for more information.

3.10 Run CEOF - Gulf Stream Interpolation

This function is the ninth option of the SAMAS menu. This function uses a driver program. This function can be run interactively or a command file can be created by the driver to run later. The command file created by the driver manages the deletion of the working or intermediate files created by the modules. The driver allows the user to select which modules of the Gulf Stream interpolation are to be executed. The source for the driver is as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Src]CEOF.C
     Vax$User1:[Peckinpaugh.SAMAS.Src]Input.C                3.2.4
```

The executable file for the driver program is as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Exe]CEOF.Exe
```

The source for the executable modules, which can be used by the command file created by this driver, are as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Src]GS_Extract.For
     Vax$User1:[Peckinpaugh.SAMAS.Src]Interp.For
     Vax$User1:[Peckinpaugh.SAMAS.Src]PrepOpto.For
     Vax$User1:[Peckinpaugh.SAMAS.Src]Opto.For
     Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For           3.2.2
     Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_LL.For          3.2.3
     Vax$User1:[Peckinpaugh.SAMAS.Src]Remove.C               3.2.5
```

The driver program creates a command file that can either be run after getting all required inputs from the user, or if the user selects, the command file is created and saved for later use. The executable modules, which can be used in the command file, are as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Exe]GS_Extract.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Interp.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]PrepOpto.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Opto.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Remove.Exe             3.2.5
```

These executables extract Gulf Stream edges from a mixed label image, create a mode file from a Gulf Stream list, prepares data for Opto, creates the interpolated Gulf Stream and a mode file for it, and deletes files.

Data files not specified by the user are required for this function. These required files include the following:

```
     Vax$User1:[Peckinpaugh.SAMAS.Dat]EigVect.Dat
     Vax$User1:[Peckinpaugh.SAMAS.Dat]Intone.Dat
     Vax$User1:[Peckinpaugh.SAMAS.Dat]Primary.Dat
     Vax$User1:[Peckinpaugh.SAMAS.Dat]Second.Dat
```

The user may have appear in his working directory any of the following working files:

```
CEOF000.Tmp
CEOF111.Tmp
CEOF222.Tmp
CEOF333.Tmp
```

These files should be deleted by the command file, unless an error occurs. If these file do exist they can be deleted.

See reference 8 for more information.

3.11 Run Hough - Eddy Detect

This is function is the tenth option of the SAMAS menu. This function consists of a driver program and the set of 4 executable modules accessed by the command file created by this driver. This function can be run interactively or a command file can be created by the driver to run later. The command file created by the driver manages the deletion of the working or intermediate files created by the modules. The driver allows the user to select which modules of the Hough eddy detection are to be executed. The source for the driver is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Hough.C
Vax$User1:[Peckinpaugh.SAMAS.Src]Input.C                    3.2.4
```

The executable file for the driver program is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Hough.Exe
```

The source for the executable modules, which can be used in the command file created by this driver, are as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Src]Edge_Extract.C
Vax$User1:[Peckinpaugh.SAMAS.Src]Dilate_Clean.For
Vax$User1:[Peckinpaugh.SAMAS.Src]EddyDetect.For
Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For               3.2.2
Vax$User1:[Peckinpaugh.SAMAS.Src]Map_XY_LL.For              3.2.3
Vax$User1:[Peckinpaugh.SAMAS.Src]Remove.C                   3.2.5
```

The driver program creates a command file that can either be run after getting all required inputs from the user, or if the user selects, the command file is created and saved for later use. The executable modules, which can be used in the command file, are as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Edge_Extract.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Dilate_Clean.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]EddyDetect.Exe
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Remove.Exe              3.2.5
```

These executables extract eddy edges from a mixed edge image, dilate the edge image, apply the Hough Transform to define eddies, and delete data files.

The user may have appear in his working directory any of the following working files:

```
     Hough00.Tmp
     Hough11.Tmp
```

These files should be deleted by the command file, unless an error occurs. If any of these files do exist they can be deleted.

3.12 Apply Image Mask

This function is the eleventh option of the SAMAS menu. This function is used to mask areas of one image with a constant value for specified values in one or more other input images. The source code required for this function is as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Src]Mask.C
     Vax$User1:[Peckinpaugh.SAMAS.Src]Input.C                 3.2.4
     Vax$User1:[Peckinpaugh.SAMAS.Src]Image_IO.For            3.2.2
```

The executable module is as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Exe]Mask.Exe
```

3.13 Vax Command

This function is the twelfth option of the SAMAS menu. This function is used to access VAX commands while still in the SAMAS menu. Not all VAX will work from the menu environment, e.g., ASSIGN. The source code required for this function is as follows:

```
     Vax$User1:[Peckinpaugh.SAMAS.Src]Vax_Command.C
     Vax$User1:[Peckinpaugh.SAMAS.Src]Input.C                 3.2.4
```

The executable module is as follows:

```
Vax$User1:[Peckinpaugh.SAMAS.Exe]Vax_Command.Exe
```

4. Discussion

The SAMAS version 1.1 is designed for easy growth and change. As new modules are created they can be easily added to the menu command file as new options. As better methods are developed for some of the existing modules, replacement is made easy by replacing records in the menu command file.

Work is currently being done to improve the performance of the expert system and labeling modules of the current system. Research is being done to develop possible replacements for the edge creation and eddy detection modules of the current system. See figure 2 for how SAMAS 1.2 is expected to look.

It is hoped that this year and next year the entire system will be transitioned from the Vax/IIS system to a SUN workstation environment.

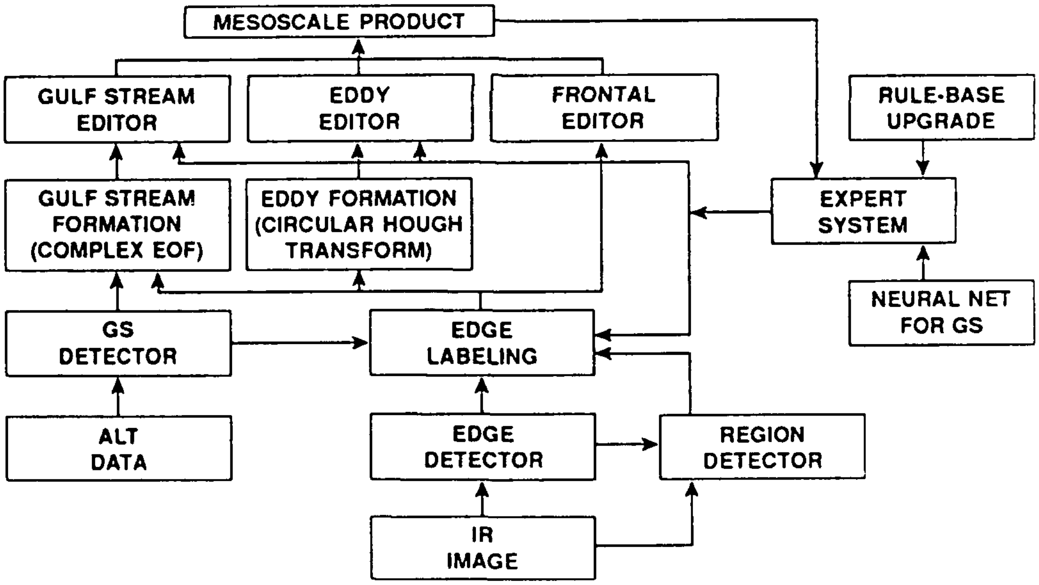

Figure 2. A functional block diagram of SAMAS 1.2.

REFERENCES

[1] *System 600 Digital Image Processing System Users Guide*, International Imaging Systems, 1500 Buckeye Drive, Milpitas, California 95035.

[2] *System 600 Digital Image Processing System Command Reference*, International Imaging Systems, 1500 Buckeye Drive, Milpitas, California 95035.

[3] M. Lybanon, J.D. McKendric, R.E. Blake, J.R.B. Cockett, and M.G. Thomason (1986). A Prototype Knowledge-Based System to Aid the Oceanographic Image Analyst, *SPIE Vol 635-Applications of Artificial Intelligence III*, pp 203-206.

[4] R.J. Holyer and S.H. Peckinpaugh (1989). Edge Detection Applied to Satellite Imagery of the Oceans, *IEEE Trans. Geosci. Remote Sensing* 27, pp 46-56.

[5] Sarah H. Peckinpaugh (1991). An Improved Method for Computing Gray-Level Cooccurrence Matrix Based Texture Measures, *CVGIP: Graphical Models and Image Processing* vol 53 no 6, pp 574-579.

[6] Theo Pavlidis (1980). A Thinning Algorithm for Discrete Binary Images, *Computer Graphics and Image Processing* 13, pp 142-157.

[7] N. Krishnakumar, S. Sitharama Iyengar, Ron Holyer, and Matthew Lybanon (1990). Feature Labelling in Infrared Oceanographic Images, *Image and Vision Computing* vol 8 no 2.

[8] E.J. Molinelli and M.J. Flanigan (1987). *Optimized CEOF Interpolation of the Gulf Stream* ,Planning Systems Inc., Tech. Rept. TR-392395, Contr. N66604-86-D-0120.

[9] *System 600 Digital Image Processing System Application Programmers Guide*, International Imaging Systems, 1500 Buckeye Drive, Milpitas, California 95035.

[10] *System 600 Digital Image Processing Subroutine Reference*, International Imaging Systems, 1500 Buckeye Drive, Milpitas, California 95035.

APPENDIX I

Example: Read/Write Gulf Stream List Files

C:

```
void Read_GS( Fname, Npts, Lat, Lon, Code )

float Lat[], Lon[];
char Fname[], Code[][2];
int *Npts;

{
   int i;
   FILE *fp;

   fp = fopen( Fname , "r");

   fscanf( fp, "%5d", Npts );

   for ( i=0; i < *Npts; i++ )
      fscanf( fp, "%10f%10f%2s", &Lat[i], &Lon[i], &Code[i] );

   fclose(fp);
}

/*-------------------------------------------------------*/

void Write_GS( Fname, Npts, Lat, Lon, Code )

float Lat[], Lon[];
char Fname[], Code[][2];
int Npts;

{
   int i;
   FILE *fp;

   fp = fopen( Fname, "w");

   fprintf( fp, "%5d\n", Npts );

   for ( i=0; i < Npts; i++ )
      fprintf( fp, "%10.4f%10.4f%2s\n", Lat[i], Lon[i],
    Code[i] );

   fclose(fp);
```

FORTRAN:

```
C-----------------------------------------------------------
C
      Subroutine Write_GS( Fname, Npts, Lat, Lon, Code )
C
C-----------------------------------------------------------
C
      character*60 Fname
      character*1 Code(*)
      Real Lat(*), Lon(*)
c
      Open( 11, File = Fname, status='New' )
c
      write(11,1) Npts
      Do i = 1, Npts
         write(11,2) lat(i), lon(i), Code(i)
      EndDo
c
      close(11)
c
 1    format(I5)
 2    format(2F10.4,A2)
      Return
      End
C
C----------------------------------------------------------
C
      subroutine Read_GS( Fname, Npts, Lat, Lon, Code )
C
C----------------------------------------------------------
C
      character*60 Fname
      character*1 Code(*)
      Real Lat(*), Lon(*)
c
      Open( 11, File = Fname, status='Old' )
c
      read(11,1) Npts
      Do i = 1, Npts
         read(11,2) lat(i), lon(i), Code(i)
      EndDo
c
      close(11)
c
 1    format(I5)
 2    format(2F10.4,A2)
      Return
      End
```

APPENDIX II

Example: Read/Write Eddy List Files

C:

```
void Read_Eddy( Fname, Npts, Lat, Lon, Rad, WC, SrcCd )

float Lat[], Lon[], Rad[];
char WC[][2], SrcCd[][3];
char Fname[];
int *Npts;

{
   int i;
   FILE *fp;

   fp = fopen( Fname, "r");

   fscanf( fp, "%5d", Npts );

   for ( i=0; i < *Npts; i++ )
      fscanf( fp, "%10f%10f%10f%2s%3s", &Lat[i], &Lon[i],
         &Rad[i], &WC[i], &SrcCd[i] );

   fclose(fp);
}

/*--------------------------------------------------------*/

void Write_Eddy( Fname, Npts, Lat, Lon, Rad, WC, SrcCd )

float Lat[], Lon[], Rad[];
char WC[][2], SrcCd[][3];
char Fname[];
int Npts;

{
   int i;
   FILE *fp;

   fp = fopen( Fname, "w");

   fprintf( fp, "%5d\n", Npts );

   for ( i=0; i < Npts; i++ )
      fprintf( fp, "%10.4f%10.4f%10.4f%2s%3s\n", Lat[i],
         Lon[i], Rad[i], WC[i], SrcCd[i] );

   fclose(fp);
}
```

FORTRAN:

```
C-------------------------------------------------------------
      Subroutine Write_Eddy( Fname, Npts, Lat, Lon, Rad, WC,
        :     SrcCd )
C-------------------------------------------------------------
C
      character*60 Fname
      Real Lat(*), Lon(*), Rad(*)
      Character*1 WC(*)
      Character*2 SrcCd(*)
c
      Open( 11, File = Fname, status='New' )
c
      write(11,1) Npts
      Do i = 1, Npts
         write(11,3) lat(i), lon(i), rad(i), WC(i), SrcCd(i)
      EndDo
c
      close(11)
c
 1    format(I5)
 3    format(3F10.4,A2,A3)
      return
      end
C
C-------------------------------------------------------------
      Subroutine Read_Eddy( Fname, Npts, Lat, Lon, Rad,
     :   WC, SrcCd )
C-------------------------------------------------------------
C
      character*60 Fname
      Real Lat(*), Lon(*), Rad(*)
      Character*1 WC(*)
      Character*2 SrcCd(*)
c
      Open( 11, File = Fname, status='Old' )
c
      Read(11,1) Npts
      Do i = 1, Npts
         read(11,3) lat(i), lon(i), rad(i), WC(i), SrcCd(i)
      EndDo
c
      close(11)
c
 1    format(I5)
 3    format(3F10.4,A2,A3)
      return
      end
```

APPENDIX III

Example: ImageIO

C:

```
#include stdio

main()
{
   float North, West, DegPixel;
   int   Ns, Nl, Fnum, Proj, Buffer[100], BuffType=3,
         Ftype, Mode, RecNum, Ierr, i, j;
   char  *Fname="Test.Image ";

   Mode = 1; /* open & read header */
   ImageIO( &Fnum, &Mode, &Ierr, Fname, &BuffType,
      &RecNum, &Ftype, &Ns, &Nl, &North, &West,
      &DegPixel, &Proj );

   printf(" %d %d %d %f %f %f %d\n", Ftype, Ns, Nl,
      North, West, DegPixel, Proj );

   /* read data records and store into I*4 buffer */
   for ( i=0, Mode=4; i < Nl; i++ ) {
      ImageIO( &Fnum, &Mode, &Ierr, Buffer, &BuffType );
      for ( j=0; j < Ns; j++ ) printf("%d ", Buffer[j]);
      printf("\n");
   }

   Mode = 6;   /* close image file */
   ImageIO( &Fnum, &Mode, &Ierr );
}
```

FORTRAN:

```
        real North, West, DegPixel
        integer Ns, Nl, Fnum, Proj, Buffer(100),
     :     BuffType, Ftype, Mode, RecNum, FileName(15)
        character*60 Fname
        equivalence (FileName, Fname)
c
        Fname = 'Test.Image'
        BuffType = 3
c
        Mode = 1          ! open and read header record
        Call ImageIO( Fnum, Mode, Ierr, FileName, BuffType,
     :     RecNum, Ftype, Ns, Nl, North, West, DegPixel, Proj )
        Write(*,*) Ftype, Ns, Nl, North, West, DegPixel, Proj
c
        Mode = 4          ! read data records
        Do i = 1, Nl
           Call ImageIO( Fnum, Mode, Ierr, Buffer, BuffType )
           write(*,'(<Ns>I5)') (Buffer(j), j=1, Ns)
        EndDo
c
        Call ImageIO(Fnum, 6, Ierr )        ! close file
c
        stop
        end
```

APPENDIX IV

Comp_Link.Com

```
$ cd Vax$User1:[Peckinpaugh.SAMAS.Exe]
$ Set Verify
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]CLUSTER_SHADE
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]CLUSTER_TEST
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]DILATE_CLEAN
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]EDDYDETECT
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]EDDYEDIT
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]EDDY_ENTERLIST
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]FRONTEDIT
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]GS_EDIT
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]GS_ENTERLIST
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]GS_EXTRACT
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]IMAGE_IO
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]INTERP
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]MAP_XY_LL
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]MEMORY_F
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]OPTO
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]PREPOPTO
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_DISPLAY
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_FORMAT
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_GRAPHICS
$ For Vax$User1:[Peckinpaugh.SAMAS.Src]THIN_LINE
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]CEOF
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]DUMP_IMAGE
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]EDGE
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]EDGE_EXTRACT
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]HOUGH
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]IIS_BE
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]IIS_DE
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]INPUT
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]MASK
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]MEMORY_C
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]REMOVE
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src]VAX_COMMAND
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src.Label]A1
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src.Label]A2
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src.Label]Label.C
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src.Label]Rdata
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Src.Label]LABEL_DRIVER
$
$ define Lnk$Library sys$library:vaxccurse.olb
$ define Lnk$Library_1 sys$library:vaxcrtlg.olb
$ define Lnk$Library_2 sys$library:vaxcrtl.olb
$ CC  Vax$User1:[Peckinpaugh.SAMAS.Screen]Driver
$ Link Driver
$
```

```
$ define Lnk$Library sys$library:vaxcrtl.olb
$
$ Link CLUSTER_SHADE, Image_IO, Memory_F, Memory_c
$ Link CLUSTER_TEST, Image_IO, Memory_F, Memory_c
$ Link DILATE_CLEAN, Image_IO, Memory_F, Memory_c
$ Link EDDYDETECT, Map_XY_ll, Image_IO, Memory_F, Memory_c
$ Link EDDY_ENTERLIST, Map_XY_LL, Image_IO
$ Link GS_ENTERLIST
$ Link GS_EXTRACT, Image_IO, Map_XY_LL
$ Link INTERP
$ Link LABEL_DRIVER, INPUT
$ Link OPTO
$ Link PREPOPTO, INTERP
$ Link THIN_LINE, Image_IO, Memory_F, Memory_c
$ Link CEOF, Input
$ Link DUMP_IMAGE, Image_IO
$ Link EDGE, Input
$ Link EDGE_EXTRACT, Image_IO
$ Link HOUGH, Input
$ Link IIS_BE
$ Link IIS_DE
$ Link MASK, Input, Image_IO
$ Link REMOVE
$ Link VAX_COMMAND, Input
$ Link Rdata, Map_XY_LL, Image_IO
$ Link A1, Image_IO, Map_XY_LL
$ Link A2, Image_IO, Map_XY_LL
$ Link Label.obj, Image_IO, Map_XY_LL
$
$ CDC Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Display
$ CDC Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Graphics
$ CDC Vax$User1:[Peckinpaugh.SAMAS.Src]SAMAS_Format
$ CDC Vax$User1:[Peckinpaugh.SAMAS.Src]EddyEdit
$ CDC Vax$User1:[Peckinpaugh.SAMAS.Src]GS_Edit
$ CDC Vax$User1:[Peckinpaugh.SAMAS.Src]FrontEdit
$
$ set noverify
$ Write sys$output "BuildExec"
$ BuildExec/obj= ( SAMAS_Display, SAMAS_Format, Image_IO, -
    Map_XY_LL, SAMAS_Graphics, EddyEdit, GS_Edit, FrontEdit, -
    Memory_f, Memory_C ) De, Be
$ Del *.Obj;*
$ Purge
$ cd last
$ cd Vax$User1:[Peckinpaugh.SAMAS.Src.Expert]
$ @cxlink.com
$ Del *.Obj;*
$ Purge
$ cd last
```

CXlink.Com

```
$ write sys$output "eddytypes"
$@Vax$User1:[Lybanon.ops83.cx1536]OC eddytypes
$ write sys$output "racycle"
$@Vax$User1:[Lybanon.ops83.cx1536]OC racycle
$ write sys$output "regions"
$@Vax$User1:[Lybanon.ops83.cx1536]OC regions
$ write sys$output "setup"
$@Vax$User1:[Lybanon.ops83.cx1536]OC setup
$ write sys$output "FinalOutput"
$@Vax$User1:[Lybanon.ops83.cx1536]OC FinalOutput
$write sys$output "wcrules"
$@Vax$User1:[Lybanon.ops83.cx1536]OC wcrules
$ write sys$output "ccrules"
$@Vax$User1:[Lybanon.ops83.cx1536]OC ccrules
$ write sys$output "eddies"
$@Vax$User1:[Lybanon.ops83.cx1536]OC eddies
$ write sys$output "mathe"
$CC mathe
$ write sys$output "nrmlz"
$CC nrmlz
$ write sys$output "movegs"
$CC movegs
$ write sys$output "graphrtn"
$CC graphrtn
$ write sys$output "IO"
$CC io
$ write sys$output "fo"
$FOR fo
$ write sys$output "purge *.obj"
$PURGE *.OBJ
$LINK/exe=EXE:Expert.Exe eddies,wcrules,ccrules,eddytypes,-
      racycle,regions,setup,mathe,nrmlz,movegs,-
      graphrtn,fo,FinalOutput,IO,-
      VAX$USER1:[LYBANON.OPS83]CRTS83,-
      VAX$USER1:[LYBANON.OPS83]MRTS83,-
      SYS$LIBRARY:VAXCRTL/LIB
```

CD Command

```
$
$ ! Command file to perform a "set default" DCL command in a simple
$ ! fashion (e.g. sd dir.dir) and verify that the specified
$ !  directory exists.
$
$ on control_y then goto terminate
$ on warning   then goto terminate
$
$ directoryOnEntry = f$environment("default")
$
$ if p1 .eqs. "" then p1 = "-"
$ dir = p1
$ log = f$trnlnm(dir)
$ if log .nes. "" then dir = log
$ len = f$length(dir)
$ if dir .eqs. "LAST" then dir = "''lastDirectory'"
$ if dir .eqs. "HOME" then dir = f$logical("sys$login")
$ if dir .eqs. "ROOT" then -
     dir = f$getdvi("sys$disk","devnam")+"[000000]"
$ if f$locate(":",dir) .eq. len .and. -
     f$locate("]",dir) .eq. len then dir := ['dir']
$
$ set def 'dir'
$ show default
$
$ lastDirectory == directoryOnEntry
$ goto update
$
$ ! Error conditions
$
$ TERMINATE:
$ set default 'directoryOnEntry'
$
$ UPDATE:
$ define/job/nolog "jobDefaultSpec" 'f$environment("default")'
$ exit
```

Distribution List

Applied Physics Laboratory
Johns Hopkins University
Johns Hopkins Road
Laurel MD 20707

Applied Physics Laboratory
University of Washington
1013 NE 40th St.
Seattle WA 98105

Applied Research Laboratory
Pennsylvania State University
P.O. Box 30
State College PA 16801-0030

Assistant Secretary of the Navy
Research, Development & Acquisition
Navy Department
Washington DC 20350-1000

Chief of Naval Operations
Navy Department
Washington DC 20350-2000
Attn: OP-71
OP-987

Chief of Naval Operations
Oceanographer of the Navy
U.S. Naval Observatory
34th & Massachusetts Ave. NW
Washington DC 20392-1800
Attn: OP-096
OP-0961B

David W. Taylor Naval Research Center
Bethesda MD 20084-5000
Attn: Commander

Defense Mapping Agency
Systems Center
8613 Lee Hwy.
Mail Stop A-13
Fairfax VA 22031-2137
Attn: Code PRN

Fleet Antisub Warfare Tng Ctr-Atl
Naval Station
Norfolk VA 23511-6495
Attn: Commanding Officer

Fleet Numerical Oceanography Center
Monterey CA 93943-5005
Attn: Commanding Officer

National Ocean Data Center
1825 Conneticut Ave., NW
Universal Bldg. South Rm. 206
Washington DC 20235

Naval Air Development Center
Warminster PA 18974-5000
Attn: Commander

Naval Air Systems Command HQ
Washington DC 20361-0001
Attn: Commander

Naval Coastal Systems Center
Panama City FL 32407-5000
Attn: Commanding Officer

Naval Facilities Engineering
Command HQ
200 Stovall St.
Alexandria VA 22332-2300
Attn: Commander

Naval Oceanographic Office
Stennis Space Center MS 39522-5001
Attn: Library (2)
Code TD, L. Bernard

Naval Oceanography Command
Stennis Space Center MS 39529-5000
Attn: Commander

Naval Oceanographic & Atmospheric
Research Laboratory
Atmospheric Directorate
Monterey CA 93943-5006
Attn: Code 400

Naval Oceanographic & Atmospheric
Research Laboratory
Stennis Space Center MS 39529-5004
Attn: Code 100
Code 105
Code 113
Code 125L (10)
Code 125P
Code 300
Code 320
Code 321 (25)

Naval Ocean Systems Center
San Diego CA 92152-5000
Attn: Commander

Naval Postgraduate School
Monterey CA 93943
Attn: Superintendent

Naval Research Laboratory
Washington DC 20375
Attn: Library

Naval Sea Systems Command HQ
Washington DC 20362-5101
Attn: Commander

Naval Surface Warfare Center Det
Silver Spring
White Oak Laboratory
10901 New Hampshire Ave.
Silver Spring MD 20903-5000
Attn: Officer in charge
Library

Naval Surface Warfare Center
Dahlgren VA 22448-5000
Attn: Commander

Naval Underwater Systems Center
Newport RI 02841-5047
Attn: Commander

Naval Underwater Systems Center Det
New London Laboratory
New London CT 06320
Attn: Officer in Charge

Office of Naval Research
800 N. Quincy St.
Arlington VA 22217-5000
Attn: Code 10D/10P, Dr. E. Silva
Code 112, Dr. E. Hartwig
Code 12
Code 10

Office of Naval Research
ONR European Office
PSC 802 Box 39
FPO AE 09499-0700
Attn: Commanding Officer

Office of Naval Technology
800 N. Quincy St.
Arlington, VA 22217-5000
Attn: Code 20, Dr. P. Selwyn
Code 228, Dr. M. Briscoe
Code 234, Dr. C. Votaw

Scripps Institution of Oceanography
University of California
291Rosecrans St.
San Diego CA 92106-3505

Space & Naval Warfare Sys Com
Director of Navy Laboratories
SPAWAR 005
Washington DC 20363-5100
Attn: Commander

Woods Hole Oceanographic Institution
P.O. Box 32
Woods Hole MA 02543
Attn: Director

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. Agency Use Only *(Leave blank).*

2. Report Date.
November 1991

3. Report Type and Dates Covered.
Final

4. Title and Subtitle.
Documentation for the Semi-Automated Mesoscale Analysis System 1.1

5. Funding Numbers.

Program Element No.	0602534N
Project No.	3582
Task No.	MOG
Accession No.	DN256010
Work Unit No.	13212R

6. Author(s).
S. H. Peckinpaugh

7. Performing Organization Name(s) and Address(es).
Naval Oceanographic and Atmospheric Research Laboratory
Ocean Science Directorate
Stennis Space Center, Mississippi 39529-5004

8. Performing Organization Report Number.
NOARL Technical Note 193

9. Sponsoring/Monitoring Agency Name(s) and Address(es).
Office of Naval Technology
800 North Quincy Street
Arlington, VA 22217-5000

10. Sponsoring/Monitoring Agency Report Number.
NOARL Technical Note 193

11. Supplementary Notes.

12a. Distribution/Availability Statement.
Approved for public release; distribution is unlimited.

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

Several Software modules have been developed for or by the Naval Oceanographic and Atmospheric Research Laboratory to automate the analysis and interpretation of satellite infrared imagery in the Gulf Stream region of the Northwest Atlantic Ocean. The modules have been integrated to form the Semi-Automated Mesoscale Analysis System Version 1.1 (SAMAS). This document provides the information required for an analyst to use SAMAS 1.1. This document provides descriptions of how the modules of SAMAS 1.1 work, and the user inputs and outputs are described. This document also provides information on the location of source code and special data files required for SAMAS 1.1.

14. Subject Terms.
Remote Sensing, Artificial Intelligence, Data Assimilation, Satellite Data

15. Number of Pages.
81

16. Price Code.

17. Security Classification of Report.
Unclassified

18. Security Classification of This Page.
Unclassified

19. Security Classification of Abstract.
Unclassified

20. Limitation of Abstract.
SAR